



The information in this document is subject to change without notice and should not be construed as a commitment by Newbridge Microsystems. While reasonable precautions have been taken, Newbridge Microsystems assumes no responsibility for any errors that may appear in this document.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Newbridge Microsystems.

The acceptance of this document will be construed as an acceptance of the foregoing conditions.

Openbus™ Interface Components - Universe User Manual

Copyright 1996, Tundra Semiconductor Corporation  
All rights reserved.

Document: 891042.MD300.05

Printed in Canada

® BI-mode is a registered trademark of DY-4 Systems Inc., denoting the Bus Isolation mode

# Table of Contents

<b>1</b>	<b>General Information .....</b>	<b>1-1</b>
1.1	Introduction .....	1-1
1.2	Product Overview .....	1-1
1.3	Using This Document .....	1-3
1.4	Conventions .....	1-4
1.4.1	Signals .....	1-4
1.4.2	Symbols .....	1-4
1.4.3	Terminology .....	1-4
<b>2</b>	<b>Functional Description.....</b>	<b>2-1</b>
2.1	Architectural Overview .....	2-1
2.1.1	VMEbus Interface .....	2-3
2.1.1.1	Universe as VMEbus Slave .....	2-3
2.1.1.2	Universe as VMEbus Master .....	2-3
2.1.2	PCI Bus Interface .....	2-4
2.1.2.1	Universe as PCI Slave.....	2-4
2.1.2.2	Universe as PCI Master .....	2-4
2.1.3	Interrupter and Interrupt Handler .....	2-4
2.1.3.1	Interrupter .....	2-4
2.1.3.2	VMEbus Interrupt Handling .....	2-5
2.1.4	DMA Controller .....	2-5
2.2	VMEbus Interface .....	2-6
2.2.1	VMEbus Requester .....	2-6
2.2.1.1	Internal Arbitration for VMEbus Requests.....	2-6
2.2.1.2	Request Modes.....	2-7
2.2.1.3	VMEbus Release.....	2-8
2.2.2	Universe as VME Master .....	2-9
2.2.2.1	Addressing Capabilities .....	2-9
2.2.2.2	Data Transfer Capabilities .....	2-10

2.2.2.3	Cycle Terminations .....	2-12
2.2.3	Universe as VME Slave .....	2-12
2.2.3.1	Coupled Transfers .....	2-13
2.2.3.2	Posted Writes .....	2-14
2.2.3.3	Prefetched Block Reads .....	2-16
2.2.3.4	VMEbus Lock Commands .....	2-18
2.2.3.5	VMEbus Read Modify Writes .....	2-18
2.2.3.6	Register Accesses .....	2-18
2.2.3.7	DTACK* Rescinding .....	2-19
2.2.4	VMEbus Configuration .....	2-19
2.2.4.1	First Slot Detector .....	2-19
2.2.4.2	VMEbus Register Access at Power-up .....	2-19
2.2.5	Automatic Slot Identification .....	2-20
2.2.5.1	Auto Slot ID: VME64 Specified .....	2-20
2.2.5.2	Auto-ID: A Proprietary Tundra Method .....	2-21
2.2.6	System Controller Functions .....	2-22
2.2.6.1	System Clock Driver .....	2-22
2.2.6.2	VMEbus Arbiter .....	2-23
2.2.6.3	IACK Daisy-Chain Driver Module .....	2-23
2.2.6.4	VMEbus Time-out .....	2-24
2.2.7	BI-Mode® .....	2-24
2.3	PCI Bus Interface .....	2-25
2.3.1	PCI Cycles - Overview .....	2-25
2.3.1.1	32-Bit Versus 64-Bit PCI .....	2-25
2.3.1.2	PCI Bus Request .....	2-26
2.3.1.3	Address Phase .....	2-26
2.3.1.4	Data Transfer .....	2-28
2.3.1.5	Termination Phase .....	2-28
2.3.1.6	Parity Checking .....	2-29
2.3.2	Universe as PCI Master .....	2-30
2.3.2.1	PCI Burst Transfers .....	2-31

	2.3.2.2	Termination .....	2-32
	2.3.2.3	Parity .....	2-33
	2.3.3	Universe as PCI Slave .....	2-33
	2.3.3.1	Data Transfer .....	2-34
	2.3.3.2	Coupled Transfers .....	2-37
	2.3.3.3	Posted Writes .....	2-38
	2.3.3.4	RMW and ADOH Cycles .....	2-40
	2.3.3.5	Exclusive Accesses .....	2-41
	2.3.3.6	Terminations .....	2-43
2.4		Slave Image Programming .....	2-44
	2.4.1	VME Slave Images .....	2-44
	2.4.1.1	VMEbus Fields .....	2-45
	2.4.1.2	PCI Bus Fields .....	2-45
	2.4.1.3	Control Fields .....	2-46
	2.4.2	PCI Bus Slave Images .....	2-47
	2.4.2.1	PCI Bus Fields .....	2-48
	2.4.2.2	VMEbus Fields .....	2-48
	2.4.2.3	Control Fields .....	2-49
	2.4.3	Special PCI Slave Image .....	2-49
2.5		Bus Error Handling .....	2-52
	2.5.1	Coupled Cycles .....	2-52
	2.5.2	Decoupled Transactions .....	2-52
	2.5.2.1	Posted Writes .....	2-52
	2.5.2.2	Prefetched Reads .....	2-54
	2.5.2.3	DMA Errors .....	2-54
	2.5.2.4	Parity Errors .....	2-55
2.6		Interrupter .....	2-56
	2.6.1	The Interrupt Channel .....	2-56
	2.6.2	PCI Interrupt Generation .....	2-57
	2.6.3	VMEbus Interrupt Generation .....	2-59

2.7	Interrupt Handling.....	2-61
2.7.1	PCI Interrupt Handling.....	2-61
2.7.2	VMEbus Interrupt Handling.....	2-61
2.7.2.1	Bus Error During VME IACK Cycle.....	2-62
2.7.3	Internal Interrupt Handling.....	2-63
2.7.3.1	VME and PCI Software Interrupts.....	2-64
2.7.3.2	Software IACK Interrupt .....	2-65
2.7.3.3	VME Ownership Interrupt .....	2-65
2.7.3.4	DMA Interrupt .....	2-66
2.7.3.5	PCI and VMEbus Error Interrupts .....	2-66
2.7.4	VME64 Auto-ID .....	2-66
2.8	DMA Controller.....	2-67
2.8.1	DMA Registers Outline.....	2-67
2.8.1.1	Source and Destination Addresses.....	2-68
2.8.1.2	Transfer Size .....	2-69
2.8.1.3	Transfer Data Width .....	2-69
2.8.1.4	DMA Command Packet Pointer .....	2-70
2.8.1.5	DMA Control and Status .....	2-70
2.8.2	Direct Mode Operation.....	2-73
2.8.3	Linked-List Operation .....	2-76
2.8.3.1	Linked List Updating .....	2-80
2.8.4	FIFO Operation and Bus Ownership.....	2-82
2.8.4.1	PCI to VME Transfers .....	2-82
2.8.4.2	VME to PCI Transfers .....	2-84
2.8.5	DMA Interrupts .....	2-85
2.8.6	DMA Error Handling .....	2-86
2.9	Registers.....	2-87
2.9.1	Register Access from the PCI Bus .....	2-88
2.9.1.1	PCI Configuration Access.....	2-88
2.9.1.2	Memory or I/O Access.....	2-88
2.9.2	Register Access from the VMEbus .....	2-89

	2.9.2.1	VMEbus Register Access Image .....	2-90
	2.9.2.2	CR/CSR Accesses .....	2-92
2.10		Utility Functions .....	2-94
	2.10.1	Resets .....	2-94
	2.10.1.1	Overview of Reset Support.....	2-94
	2.10.2	Universe Reset Circuitry .....	2-96
	2.10.3	Reset Implementation Issues .....	2-97
	2.10.4	Power-Up Options.....	2-98
	2.10.4.1	Power-up Option Descriptions.....	2-99
	2.10.5	Hardware Initialization (Normal Operating Mode) .....	2-103
	2.10.6	Support for board testing.....	2-104
	2.10.6.1	Auxiliary Test Modes .....	2-104
	2.10.6.2	JTAG support.....	2-105
	2.10.7	CLK64.....	2-105
<b>3</b>		<b>Description of Signals .....</b>	<b>3-1</b>
	3.1	VMEbus Signals .....	3-1
	3.2	PCI Bus Signals .....	3-5
<b>4</b>		<b>Signals and DC Characteristics .....</b>	<b>4-1</b>
	4.1	Terminology.....	4-1
	4.2	DC Characteristics and Pin Assignments.....	4-2
<b>Appendix A – Registers .....</b>			<b>App A-1</b>
<b>Appendix B – Performance .....</b>			<b>App B-1</b>
	B.1	PCI Slave Channel .....	App B-2
	B.1.1	Coupled Cycles .....	App B-2
	B.1.1.1	Request of VMEbus.....	App B-2
	B.1.1.2	Read Cycles .....	App B-2
	B.1.1.3	Write Cycles.....	App B-3
	B.1.2	Decoupled Cycles.....	App B-4
	B.1.2.1	Write Cycles.....	App B-4

B.2	VME Slave Channel.....	App B-7
B.2.1	Coupled Cycles .....	App B-7
B.2.1.1	Block vs. non-Block Transfers .....	App B-7
B.2.1.2	Read Cycles .....	App B-7
B.2.1.3	Write Cycles.....	App B-8
B.2.2	Decoupled Cycles.....	App B-9
B.2.2.1	Write Cycles.....	App B-9
B.2.2.2	Prefetched Read Cycles .....	App B-12
B.3	DMA Channel .....	App B-13
B.3.1	VMEbus Ownership Modes .....	App B-13
B.3.2	VME Transfers.....	App B-14
B.3.2.1	Read Transfers .....	App B-14
B.3.2.2	Write Transfers .....	App B-14
B.3.3	PCI Transfers.....	App B-14
B.4	Summary .....	App B-16

## **Appendix C – Typical Applications .....App C-1**

C.1	VME Interface.....	App C-1
C.1.1	Transceivers .....	App C-1
C.1.1.1	Pull down resistors .....	App C-4
C.1.2	Direction control .....	App C-4
C.1.3	Power-up Options.....	App C-5
C.2	PCI Bus Interface .....	App C-6
C.2.1	Resets .....	App C-7
C.2.2	Local Interrupts .....	App C-8



C.3	Manufacturing Test Pins .....	App C-9
C.4	Decoupling VDD and VSS on the Universe.....	App C-9
<b>Appendix D – Reliability Prediction .....</b>		<b>App D-1</b>
D.1	Physical characteristics .....	App D-1
D.2	Thermal characteristics .....	App D-1
D.3	Universe Ambient Operating Calculations .....	App D-2
D.4	Thermal vias.....	App D-2
<b>Appendix E – Cycle Mapping .....</b>		<b>App E-1</b>
E.1	Little-endian Mode.....	App E-1
<b>Appendix F – Why and How to Use BGA Packaging .....</b>		<b>App F-1</b>
F.1	Comparing BGA to Other Surface Mount Packages .....	App F-1
F.2	Anatomy of the PBGA .....	App F-4
F.2.1	Bringing the BGA to Your Circuit Board .....	App F-5
F.3	Rework and Repair.....	App F-10
F.4	Moisture Effects and Handling .....	App F-12
F.5	Conclusion .....	App F-12
<b>Appendix G – Operating and Storage Conditions .....</b>		<b>App G-1</b>
<b>Appendix H – Mechanical and Ordering Information .....</b>		<b>App H-1</b>
H.1	Mechanical Information .....	App H-1
H.2	Ordering Information .....	App H-3
<b>Index .....</b>		<b>Index-1</b>



# List of Tables

Table 1.1 :	Signal Conventions Used in this Manual .....	1-4
Table 2.2 :	Command Type Encoding for Transfer Type.....	2-27
Table 2.3 :	Register Fields for the Special Cycle Generator .....	2-40
Table 2.4 :	VMEbus Fields for VME Slave Image.....	2-44
Table 2.5 :	PCI Bus Fields for VME Slave Image.....	2-44
Table 2.6 :	Control Fields for VME Slave Image.....	2-44
Table 2.7 :	PCI Bus Fields for the PCI Bus Slave Image .....	2-47
Table 2.8 :	VMEbus Fields for the PCI Bus Slave Image .....	2-47
Table 2.9 :	Control Fields for PCI Bus Slave Image .....	2-47
Table 2.10 :	PCI Bus Fields for the Special PCI Slave Image.....	2-49
Table 2.11 :	VMEbus Fields for the Special PCI Bus Slave Image .....	2-50
Table 2.12 :	Control Fields for the Special PCI Bus Slave Image.....	2-50
Table 2.13 :	PCI Interrupt Source Enabling, Mapping, and Status Bits .....	2-58
Table 2.14 :	VMEbus Interrupt Source Enabling, Mapping, and Status Bits.....	2-60
Table 2.15 :	Internal Interrupt Routing.....	2-63
Table 2.16 :	DMA Interrupt Sources and Enable Bits.....	2-85
Table 2.17 :	Programming the VMEbus Register Access Image .....	2-90
Table 2.18 :	Universe Signals Involved in Reset Circuitry .....	2-94
Table 2.19 :	Universe Register Support for Resets.....	2-95
Table 2.20 :	VRAI Base Address Power-up Options .....	2-100
Table 2.21 :	Power-Up Options .....	2-102
Table 2.22 :	Manufacturing Pin Requirements for Normal Operating Mode.....	2-103
Table 2.23 :	Test Mode Operation.....	2-104
Table 4.1	DC Electrical Characteristics ( $V_{DD} = 5\text{ V} \pm 10\%$ ).....	4-2
Table 4.2	Pin List and DC Characteristics for Universe Signals <sup>1</sup> .....	4-3
Table 4.3	PCI Bus Address/Data Pins .....	4-7
Table 4.4	VMEbus Address Pins.....	4-9
Table 4.5	VMEbus Data Pins .....	4-10

Table 4.6	Pin Assignments for Power and Ground .....	4-11
Table A.1 :	Universe Register Map .....	App A-2
Table A.2 :	PCI Configuration Space ID Register (PCI_ID) .....	App A-6
Table A.3 :	PCI Configuration Space Control and Status Register (PCI_CSR) App A-7	
Table A.4 :	PCI Configuration Class Register (PCI_CLASS) .....	App A-9
Table A.5 :	PCI Configuration Miscellaneous 0 Register (PCI_MISC0) .....	App A-10
Table A.6 :	PCI Configuration Base Address Register (PCI_BS).....	App A-11
Table A.7 :	PCI Configuration Miscellaneous 1 Register (PCI_MISC1) .....	App A-12
Table A.8 :	PCI Slave Image 0 Control (LSI0_CTL).....	App A-13
Table A.9 :	PCI Slave Image 0 Base Address Register (LSI0_BS) .....	App A-14
Table A.10 :	PCI Slave Image 0 Bound Address Register (LSI0_BD).....	App A-15
Table A.11 :	PCI Slave Image 0 Translation Offset (LSI0_TO).....	App A-16
Table A.12 :	PCI Slave Image 1 Control (LSI1_CTL).....	App A-17
Table A.13 :	PCI Slave Image 1 Base Address Register (LSI1_BS) .....	App A-18
Table A.14 :	PCI Slave Image 1 Bound Address Register (LSI1_BD).....	App A-19
Table A.15 :	PCI Slave Image 1 Translation Offset (LSI1_TO).....	App A-20
Table A.16 :	PCI Slave Image 2 Control (LSI2_CTL).....	App A-21
Table A.17 :	PCI Slave Image 2 Base Address Register (LSI2_BS) .....	App A-22
Table A.18 :	PCI Slave Image 2 Bound Address Register (LSI2_BD).....	App A-23
Table A.19 :	PCI Slave Image 2 Translation Offset (LSI2_TO).....	App A-24
Table A.20 :	PCI Slave Image 3 Control (LSI3_CTL).....	App A-25
Table A.21 :	PCI Slave Image 3 Base Address Register (LSI3_BS) .....	App A-26
Table A.22 :	PCI Slave Image 3 Bound Address Register (LSI3_BD).....	App A-27
Table A.23 :	PCI Slave Image 3 Translation Offset (LSI3_TO).....	App A-28
Table A.24 :	Special Cycle Control Register (SCYC_CTL).....	App A-29
Table A.25 :	Special Cycle PCI bus Address Register .....	App A-30
Table A.26 :	Special Cycle Swap/Compare Enable Register (SCYC_EN).....	App A-31
Table A.27 :	Special Cycle Compare Data Register (SCYC_CMP).....	App A-32
Table A.28 :	Special Cycle Swap Data Register (SCYC_SWP) .....	App A-33
Table A.29 :	PCI Miscellaneous Register (LMISC).....	App A-34

Table A.30 :	Special PCI Slave Image (SLSI) .....	App A-35
Table A.31 :	PCI Command Error Log Register (L_CMDERR) .....	App A-37
Table A.32 :	PCI Address Error Log (LAERR) .....	App A-38
Table A.33 :	DMA Transfer Control Register (DCTL).....	App A-39
Table A.34 :	DMA Transfer Byte Count Register (DTBC) .....	App A-40
Table A.35 :	DMA PCI Bus Address Register (DLA) .....	App A-41
Table A.36 :	DMA VMEbus Address Register (DVA).....	App A-42
Table A.37 :	DMA Command Packet Pointer (DCPP) .....	App A-43
Table A.38 :	DMA General Control/Status Register (DGCS) .....	App A-44
Table A.39 :	DMA Linked List Update Enable Register (D_LLUE).....	App A-46
Table A.40 :	PCI Interrupt Enable Register (LINT_EN) .....	App A-47
Table A.41 :	PCI Interrupt Status Register (LINT_STAT) .....	App A-49
Table A.42 :	PCI Interrupt Map 0 Register (LINT_MAP0).....	App A-51
Table A.43 :	PCI Interrupt Map 1 Register (LINT_MAP1).....	App A-52
Table A.44 :	VMEbus Interrupt Enable Register (VINT_EN).....	App A-53
Table A.45 :	VMEbus Interrupt Status Register (VINT_STAT) .....	App A-54
Table A.46 :	VME Interrupt Map 0 Register (VINT_MAP0).....	App A-55
Table A.47 :	VME Interrupt Map 1 Register (VINT_MAP1).....	App A-56
Table A.48 :	Interrupt STATUS/ID Out (STATID).....	App A-57
Table A.49 :	VIRQ1 STATUS/ID Register (V1_STATID).....	App A-58
Table A.50 :	VIRQ2 STATUS/ID Register (V2_STATID).....	App A-59
Table A.51 :	VIRQ3 STATUS/ID Register (V3_STATID).....	App A-60
Table A.52 :	VIRQ4 STATUS/ID Register (V4_STATID).....	App A-61
Table A.53 :	VIRQ5 STATUS/ID Register (V5_STATID).....	App A-62
Table A.54 :	VIRQ6 STATUS/ID Register (V6_STATID).....	App A-63
Table A.55 :	VIRQ7 STATUS/ID Register (V7_STATID).....	App A-64
Table A.56 :	Master Control Register (MAST_CTL) .....	App A-65
Table A.57 :	Miscellaneous Control Register (MISC_CTL).....	App A-67
Table A.58 :	Miscellaneous Status Register (MISC_STAT).....	App A-69
Table A.59 :	User AM Codes Register (USER_AM).....	App A-70
Table A.60 :	VMEbus Slave Image 0 Control (VSI0_CTL).....	App A-71

Table A.61 :	VMEbus Slave Image 0 Base Address Register (VSI0_BS).....	App A-72
Table A.62 :	VMEbus Slave Image 0 Bound Address Register (VSI0_BD) ..	App A-73
Table A.63 :	VMEbus Slave Image 0 Translation Offset (VSI0_TO) .....	App A-74
Table A.64 :	VMEbus Slave Image 1 Control (VSI1_CTL).....	App A-75
Table A.65 :	VMEbus Slave Image 1 Base Address Register (VSI1_BS).....	App A-76
Table A.66 :	VMEbus Slave Image 1 Bound Address Register (VSI1_BD) ..	App A-77
Table A.67 :	VMEbus Slave Image 1 Translation Offset (VSI1_TO) .....	App A-78
Table A.68 :	VMEbus Slave Image 2 Control (VSI2_CTL).....	App A-79
Table A.69 :	VMEbus Slave Image 2 Base Address Register (VSI2_BS).....	App A-80
Table A.70 :	VMEbus Slave Image 2 Bound Address Register (VSI2_BD) ..	App A-81
Table A.71 :	VMEbus Slave Image 2 Translation Offset (VSI2_TO) .....	App A-82
Table A.72 :	VMEbus Slave Image 3 Control (VSI3_CTL).....	App A-83
Table A.73 :	VMEbus Slave Image 3 Base Address Register (VSI3_BS).....	App A-84
Table A.74 :	VMEbus Slave Image 3 Bound Address Register (VSI3_BD) ..	App A-85
Table A.75 :	VMEbus Slave Image 3 Translation Offset (VSI3_TO) .....	App A-86
Table A.76 :	VMEbus Register Access Image Control Register (VRAI_CTL) .....	App A-87
Table A.77 :	VMEbus Register Access Image Base Address Register (VRAI_BS) .....	App A-88
Table A.78 :	VMEbus CSR Control Register (VCSR_CTL).....	App A-89
Table A.79 :	VMEbus CSR Translation Offset (VCSR_TO).....	App A-90
Table A.80 :	VMEbus AM Code Error Log (V_AMERR) .....	App A-91
Table A.81 :	VMEbus Address Error Log (VAERR).....	App A-92
Table A.82 :	VMEbus CSR Bit Clear Register (VCSR_CLR) .....	App A-93
Table A.83 :	VMEbus CSR Bit Set Register (VCSR_SET).....	App A-94
Table A.84 :	VMEbus CSR Base Address Register (VCSR_BS) .....	App A-95
Table B.1 :	PCI Slave Channel Performance .....	App B-16
Table B.2 :	VME Slave Channel Performance.....	App B-17
Table B.3 :	DMA Channel Performance .....	App B-18
Table B.4 :	Daisy Chains.....	App B-18
Table C.1 :	VMEbus Signal Drive Strength Requirements.....	App C-4
Table C.2 :	Reset Signals.....	App C-7
Table D.1 :	θja Values For Various Air Flows and Package Types.....	App D-2

Table E.1 :	Mapping of 32-bit Little-Endian PCI Bus to 32-bit VMEbus .....	App E-1
Table E.2 :	Mapping of 32-bit Little-Endian PCI Bus to 64-bit VMEbus .....	App E-2
Table F.1 :	Parameters affecting escape routing .....	App F-7
Table G.1 :	Recommended Operating Conditions .....	App G-1
Table G.2 :	Absolute Maximum Ratings .....	App G-1





# List of Figures

Figure 2.1 :	Architectural Diagram for the Universe .....	2-2
Figure 2.2 :	Influence of Transaction Data Width and Slave Image Data Width on Data Packing/Unpacking.....	2-11
Figure 2.3 :	VMEbus Slave Channel Dataflow.....	2-13
Figure 2.4 :	Timing for Auto-ID Cycle.....	2-22
Figure 2.5 :	PCI bus Slave Channel Dataflow .....	2-35
Figure 2.6 :	Influence of Transaction Data Width and Slave Image Data Width on Data Packing/Unpacking.....	2-36
Figure 2.7 :	Address Translation Mechanism for VMEbus to PCI Bus Transfers .	2-46
Figure 2.8 :	Address Translation Mechanism for PCI Bus to VMEbus Transfers .	2-48
Figure 2.9 :	Memory Mapping in the Special PCI Slave Image .....	2-51
Figure 2.10 :	Interrupt Sources and Destinations.....	2-57
Figure 2.11 :	STATUS/ID Provided by Universe.....	2-60
Figure 2.12 :	Universe Sources of Internal Interrupts.....	2-64
Figure 2.13 :	Direct Mode DMA transfers.....	2-74
Figure 2.14 :	Command Packet Structure and Linked List Operation.....	2-77
Figure 2.15 :	DMA Linked List Operation .....	2-78
Figure 2.16 :	Universe Control and Status Register Space .....	2-87
Figure 2.17 :	PCI Bus Access to UCSR as Memory or I/O Space.....	2-89
Figure 2.18 :	UCSR Access from the VMEbus Register Access Image.....	2-91
Figure 2.19 :	UCSR Access in VMEbus CR/CSR Space .....	2-93
Figure 2.20 :	Functional Block Diagram of Reset Circuitry.....	2-96
Figure 2.21 :	PWRRST# assertion.....	2-98
Figure 2.22 :	Power-up Options Timing .....	2-99
Figure 4. 1:	Pinout for 313-pin Plastic BGA Package .....	4-12
Figure 4. 2:	Pinout for 324-pin Ceramic BGA Package.....	4-13
Figure A.1 :	UCSR Access Mechanisms .....	App A-1
Figure B.1 :	Coupled Read Cycle - Universe as VME Master .....	App B-3

Figure B.2 :	Several Coupled Read Cycles - Universe as VME Master .....	App B-3
Figure B.3 :	Coupled Write Cycle - Universe as VME Master .....	App B-4
Figure B.4 :	Several Non-Block Decoupled Writes - Universe as VME Master .....	App B-6
Figure B.5 :	BLT Decoupled Write - Universe as VME Master .....	App B-6
Figure B.6 :	Coupled Read Cycle - Universe as VME Slave .....	App B-8
Figure B.7 :	Coupled Write Cycle - Universe as VME Slave .....	App B-9
Figure B.8 :	Non-Block Decoupled Write Cycle - Universe as VME Slave...	App B-10
Figure B.9 :	BLT Decoupled Write Cycle - Universe as VME Slave .....	App B-11
Figure B.10 :	MBLT Decoupled Write Cycle - Universe as VME Slave .....	App B-11
Figure B.11 :	BLT Pre-fetched Read Cycle - Universe as VME Slave.....	App B-13
Figure B.12 :	PCI Read Transactions During DMA Operation.....	App B-15
Figure C.1 :	Universe Connections to the VMEbus Through TTL Buffers .....	App C-2
Figure C.2 :	Power-up Configuration Using Passive Pull-ups .....	App C-5
Figure C.3 :	Power-up Configuration Using Active Circuitry.....	App C-6
Figure C.4 :	Analog Isolation Scheme.....	App C-9
Figure C.5 :	Noise Filter Scheme.....	App C-10
Figure F.1 :	Footprint and height of BGA.....	App F-2
Figure F.2 :	Thermal characteristics of QFP vs. OMPAC .....	App F-3
Figure F.3 :	Electrical performance of various packages.....	App F-4
Figure F.4 :	Cross-Section of the PBGA.....	App F-5
Figure F.5 :	Land Design Patterns.....	App F-6
Figure H.1 :	Mechanical Dimensions for the 324-Pin Ceramic BGA Package	App H-1
Figure H.2 :	Mechanical Dimensions for 313-Pin Plastic BGA Package.....	App H-2

# 1 General Information

## 1.1 Introduction

The Universe VMEbus interface chip (CA91C042) provides a reliable high performance 64-bit VMEbus to PCI interface in one device. The Universe device results from our ongoing commitment to Openbus interface design, which has produced the SCV64 and AVICS VMEbus interfaces, and the LIFE Futurebus+ interface.

Designed in consultation with Motorola, the Universe is compliant with the VME64 specification and is tuned to the new generation of high speed processors.

The Universe is ideally suited for CPU boards acting as both master and slave in the VMEbus system, and is particularly fitted for PCI local systems. The Universe is manufactured in a CMOS process.

## 1.2 Product Overview

### The Features

- Fully compliant, 64-bit, 33 MHz PCI local bus interface,
- fully compliant, high performance 64-bit VMEbus interface,
- integral FIFOs for write posting to maximize bandwidth utilization,
- programmable DMA controller with linked list support,
- VMEbus transfer rates of 60-70 MBytes/sec,
- complete suite of VMEbus address and data transfer modes,
  - A32/A24/A16 master and slave
  - D64 (MBLT)/D32/D16/D08 master and slave
  - BLT, ADOH, RMW, LOCK
- automatic initialization for slave-only applications,
- flexible register set, programmable from both the PCI bus and VMEbus ports,
- full VMEbus system controller functionality,
- IEEE 1149.1 JTAG testability support, and
- available in 313-pin Plastic BGA and 324-pin contact Ceramic BGA.

## Industry Proven Performance

The Universe builds on Tundra's experience with its highest performing, commercially available single chip VME64 interface, the SCV64. This performance excellence is the product of the decoupling architecture unique to Tundra's family of Bus-Bridging Components. The Universe uses built-in FIFOs to decouple the transfer of data between the VMEbus and PCI bus, thus compensating for mismatches in relative bus performance. In addition to the advantage of decoupling, the Universe offers block transfer capability (including BLT and MBLT), and an integral DMA controller.

## Industry Proven VMEbus Attributes

Like the SCV64, the Universe implements all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to support 3U applications, i.e. A40 and MD32). In addition, it provides full VMEbus system controller functionality with multilevel arbitration modes. All VMEbus features are fully programmable from the PCI and VMEbus side, making the Universe a flexible interface solution. With its other VMEbus functions, the Universe provides full VMEbus interrupter and interrupt handler capability, automatic system controller logic, and multilevel VMEbus requester. The Universe also supports read-modify-write operations on the VMEbus.

## The Latest Surface Mount Technology

The Universe offers tremendous board area savings through the use of BGA packaging technology. The Universe ceramic BGA package is solder bumped with a contact array pitch of 0.050 inches. This packaging approach effectively yields 'chip on board' densities using surface mount technology.

## PCI: The *de facto* Local Bus Interface Standard

As a result of the computing industry's widespread acceptance of PCI, the leading semiconductor vendors have built PCI support into their newest processor and peripheral families. To allow VMEbus single board computer vendors to capitalize on these new PCI components, Tundra has developed a PCI to VMEbus interface controller, the Universe. The availability of the Universe, with PCI support, eases the development of multi-master, multi-processor architectures on VMEbus systems.

## 1.3 Using This Document

The manual is organized as follows:

- Chapter 1 - General Information,
- Chapter 2 - Functional Description,
- Chapter 3 - Signal Description,
- Chapter 4 - Signals and DC Characteristics,
- Appendix A - Registers,
- Appendix B- Performance,
- Appendix C- Typical Applications,
- Appendix D- Reliability,
- Appendix E- Cycle Mapping,
- Appendix F- BGA Packaging,
- Appendix G- Operating and Storage Conditions
- Appendix H- Mechanical and Ordering Information, and
- Index.

Chapter 1 introduces the Universe and provides the reader with information about the necessary concepts and conventions required to use the manual.

Chapters 2 to 4 describe Universe function, beginning with overall functionality in Chapter 2 to detailed signal descriptions in Chapters 3 and 4.

The Appendices are reference sources necessary for the implementation of the Universe. In addition, the Appendices contain application information to aid the user in system design.

The Index provides a means to quickly access information on a keyword basis.

## 1.4 Conventions

### 1.4.1 Signals

Signals on the VMEbus and PCI bus are either active high or active low. Active low signals are defined as true (asserted) when they are at logic low. Similarly, active high signals are defined as true at a logic high. Signals are considered asserted when active and negated when inactive, irrespective of voltage levels.

For voltage levels, the use of ‘0’ indicates a low voltage while a ‘1’ indicates a high voltage.

**Table 1.1 : Signal Conventions Used in this Manual**

SIGNAL#	active low signals
---------	--------------------

### 1.4.2 Symbols



*Caution: This symbol alerts the reader to procedures or operating levels which may result in misuse of or damage to the Universe.*



*Note: This symbol directs the reader's attention to useful information or suggestions.*

### 1.4.3 Terminology

When discussing bus ownership, this manual uses “master” to indicate the bus owner, and “slave” to indicate the address accessed by the master. These terms are used for both PCI and VMEbus descriptions. For PCI bus discussions, the reader should be aware that “master” and “slave” correspond to “initiator” and “target” in PCI terminology.

Note also that the term “cycle” refers to a single data beat, while a “transaction” is composed of one or more “cycles”.

## 2 Functional Description

### 2.1 Architectural Overview

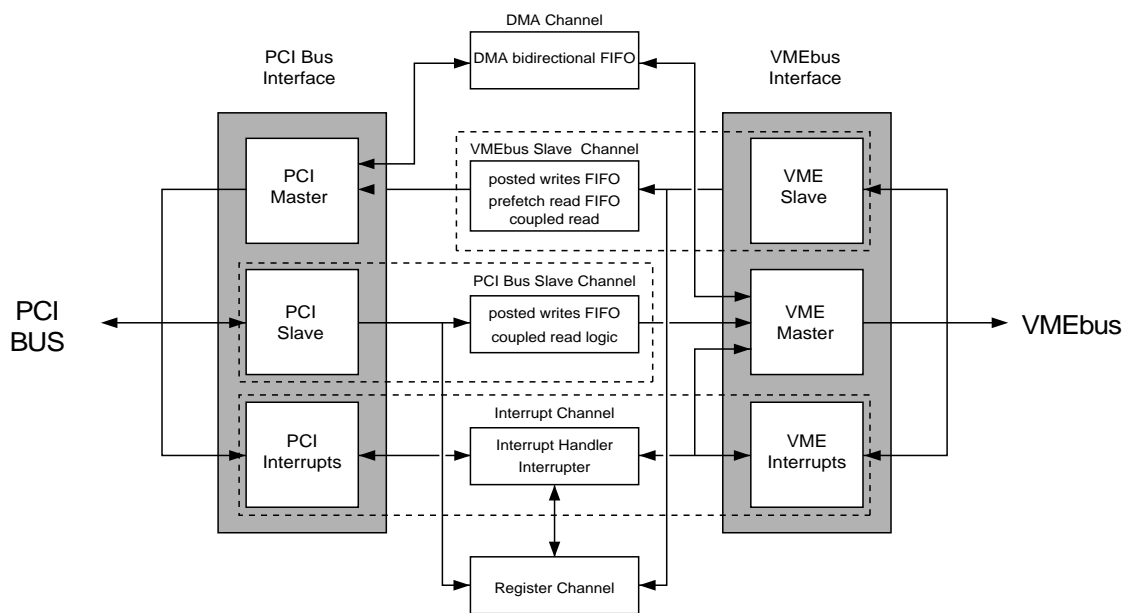
This section introduces the general architecture of the Universe. This description makes frequent reference to the functional block diagram provided in Figure 2.1 on page 2-2. Notice that for each of the interfaces, VMEbus and PCI bus, there are three functionally distinct modules: master module, slave module, and interrupt module. These modules are connected to the different functional channels operating in the Universe. These channels are:

- the VME Slave Channel,
- the PCI Bus Slave Channel,
- the DMA Channel,
- the Interrupt Channel, and
- the Register Channel.

The Architectural Overview is organized into the following sections:

- “VMEbus Interface”,
- “PCI Bus Interface”,
- “Interrupter and Interrupt Handler”, and
- “DMA Controller”.

These sections describe the operation of the Universe in terms of the different modules and channels illustrated in Figure 2.1.



**Figure 2.1 : Architectural Diagram for the Universe**



## 2.1.1 VMEbus Interface

### 2.1.1.1 Universe as VMEbus Slave

The Universe VME Slave Channel accepts all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to support 3U applications, i.e. A40 and MD32). Incoming write transactions from the VMEbus may be treated as either coupled or posted, depending upon the programming of the VMEbus slave image (see “VME Slave Images” on page 2-44). With posted write transactions, data is written to a Posted Write Receive FIFO (RXFIFO), and the VMEbus master receives data acknowledgment from the Universe. Write data is transferred to the PCI resource from the RXFIFO without the involvement of the initiating VMEbus master (see “Posted Writes” on page 2-14 for a full explanation of this operation). With a coupled cycle, the VMEbus master only receives data acknowledgment when the transaction is complete on the PCI bus. This means that the VMEbus is unavailable to other masters while the PCI bus transaction is executed.

Read transactions may be either prefetched or coupled. If enabled by the user, a prefetched read is initiated when a VMEbus master requests a block read transaction (BLT or MBLT) and this mode is enabled. When the Universe receives the block read request, it begins to fill its Read Data FIFO (RDFIFO) using burst transactions from the PCI resource. The initiating VMEbus master then acquires its block read data from the RDFIFO rather than from the PCI resources directly.

### 2.1.1.2 Universe as VMEbus Master

The Universe becomes VMEbus master when the VME Master Interface is internally requested by the PCI Bus Slave Channel, the DMA Channel, or the Interrupt Channel. The Interrupt Channel always has priority over the other two channels. Several mechanisms are available to configure the relative priority that the PCI Bus Slave Channel and DMA Channel have over ownership of the VMEbus Master Interface.

The Universe’s VME Master Interface generates all of the addressing and data transfer modes documented in the VME64 specification (except A64 and those intended to support 3U applications, i.e. A40 and MD32). The Universe is also compatible with all VMEbus modules conforming to pre-VME64 specifications. As VMEbus master, the Universe supports Read-Modify-Write (RMW), and Address-Only-with-Handshake (ADOH) but does not accept RETRY\* as a termination from the VMEbus slave. The ADOH cycle is used to implement the VMEbus Lock command allowing a PCI master to lock VMEbus resources.

## **2.1.2 PCI Bus Interface**

### **2.1.2.1 Universe as PCI Slave**

Read transactions from the PCI bus are always processed as coupled. Write transactions may be either coupled or posted, depending upon the setting of the PCI bus slave image (see “PCI Bus Slave Images” on page 2-47). With a posted write transaction, write data is written to a Posted Write Transmit FIFO (TXFIFO) and the PCI bus master receives data acknowledgment from the Universe with zero wait states. Meanwhile, the Universe obtains the VMEbus and writes the data to the VMEbus resource independent of the initiating PCI master (see “Posted Writes” on page 2-38 for a full description of this operation).

To allow PCI masters to perform RMW and ADOH cycles, the Universe provides a Special Cycle Generator. The Special Cycle Generator can be used in combination with a VMEbus ownership function to guarantee PCI masters exclusive access to VMEbus resources over several VMEbus transactions (see “Exclusive Accesses” on page 2-41 and “RMW and ADOH Cycles” on page 2-40 for a full description of this functionality).

### **2.1.2.2 Universe as PCI Master**

The Universe becomes PCI master when the PCI Master Interface is internally requested by the VME Slave Channel or the DMA Channel. There are mechanisms provided which allow the user to configure the relative priority of the VME Slave Channel and the DMA Channel.

## **2.1.3 Interrupter and Interrupt Handler**

### **2.1.3.1 Interrupter**

The Universe interrupt channel provides a flexible scheme to map interrupts to either the PCI bus or VMEbus interface. Interrupts are generated from either hardware or software sources (see “Interrupter” on page 2-56 for a full description of hardware and software sources).

Interrupt sources can be mapped to any of the PCI bus or VMEbus interrupt output pins. Interrupt sources mapped to VMEbus interrupts are generated on the VMEbus interrupt output pins VIRQ# [7:1]. When a software and hardware source are assigned to the same VIRQn# pin, the software source always has higher priority.

Interrupt sources mapped to PCI bus interrupts are generated on one of the INT# [7:0] pins. To be fully PCI compliant, all interrupt sources must be routed to a single INT# pin.

For VMEbus interrupt outputs, the Universe interrupter supplies an 8-bit STATUS/ID to a VMEbus interrupt handler during the IACK cycle, and optionally generates an internal interrupt to signal that the interrupt vector has been provided (see “VMEbus Interrupt Generation” on page 2-59).

Interrupts mapped to PCI bus outputs are serviced by the PCI interrupt controller. The CPU determines which interrupt sources are active by reading an interrupt status register in the Universe. The source negates its interrupt when it has been serviced by the CPU (see “PCI Interrupt Generation” on page 2-57).

### 2.1.3.2 VMEbus Interrupt Handling

A VMEbus interrupt triggers the Universe to generate a normal VMEbus IACK cycle and generate the specified interrupt output. When the IACK cycle is complete, the Universe releases the VMEbus and the interrupt vector is read by the PCI resource servicing the interrupt output. Software interrupts are ROAK, while hardware, and internal interrupts are RORA.

### 2.1.4 DMA Controller

The Universe provides an internal DMA controller for high performance data transfer between the PCI and VMEbus. DMA operations between the source and destination bus are decoupled through the use of a single bidirectional FIFO (DMAFIFO). Parameters for the DMA transfer are software configurable in the Universe registers (see “DMA Controller” on page 2-67).

The principal mechanism for DMA transfers is the same for operations in either direction (PCI to VME, or VME to PCI), only the relative identity of the source and destination bus changes. In a DMA transfer, the Universe gains control of the source bus and reads data into its DMAFIFO. Following specific rules of DMAFIFO operation (see “FIFO Operation and Bus Ownership” on page 2-82), it then acquires the destination bus and writes data from its DMAFIFO.

The DMA controller can be programmed to perform multiple blocks of transfers using entries in a linked-list. The DMA will work through the transfers in the linked-list following pointers at the end of each linked-list entry. Linked-list operation is initiated through a pointer in an internal Universe register, but the linked-list itself resides in PCI bus memory.

## 2.2 VMEbus Interface

The VMEbus interface incorporates all operations associated with the VMEbus. This includes master and slave functions, VMEbus configuration and system controller functions. These operations are covered as follows:

- “VMEbus Requester” below,
- “Universe as VME Master” on page 2-9,
- “Universe as VME Slave” on page 2-12,
- “VMEbus Configuration” on page 2-19,
- “Automatic Slot Identification” on page 2-20, and
- “System Controller Functions” on page 2-22.

For information concerning the Universe VMEbus slave images, see “VME Slave Images” on page 2-44.

### 2.2.1 VMEbus Requester

#### 2.2.1.1 Internal Arbitration for VMEbus Requests

Three different internal channels within the Universe require use of the VMEbus: the Interrupt Channel, the PCI Slave Channel, and the DMA Channel. These three channels do not directly request the VMEbus, instead they compete internally for ownership of the VME Master Interface.

The Interrupt Channel (refer to Figure 2.1 on page 2-2) always has the highest priority for access to the VME Master Interface. The DMA and PCI Slave Channel requests are handled in a round robin fashion. The channel awarded VMEbus mastership maintains ownership of the VMEbus until it is ‘done’. The definition of ‘done’ for each channel is given below in “VMEbus Release” on page 2-8.

The Interrupt Channel requests the VMEbus master when it detects an enabled VMEbus interrupt line asserted and needs to run an interrupt acknowledge cycle to acquire the STATUS/ID.

The PCI Slave Channel requests the VME Master Interface to service the following conditions:

- the TXFIFO contains a complete transaction, or
- if there is a coupled cycle request.

The DMA Channel requests the VME Master Interface if:

- the DMAFIFO level reaches the programmed PCI aligned burst size (as set with PABS in the MAST\_CTL register), or
- the DMA block is complete (see “DMA Controller” on page 2-67).

In the case of the DMA Channel, the user can optionally use the DMA Channel VMEbus-off-timer to further qualify requests from this channel. The VMEbus-off-timer controls how long the DMA remains off the VMEbus before making another request (see “PCI to VME Transfers” on page 2-82).

The Universe provides a software mechanism for VMEbus acquisition through the VMEbus ownership bit (VOWN in the MAST\_CTL register, Table A.56). When the VMEbus ownership bit is set, the Universe acquires the VMEbus and sets an acknowledgment bit (VOWN\_ACK in the MAST\_CTL register, Table A.56) and optionally generates an interrupt to the PCI bus (see “Exclusive Accesses” on page 2-41). The Universe maintains VMEbus ownership until the ownership bit is cleared. During the VMEbus tenure initiated by setting the ownership bit, only the PCI Slave Channel and Interrupt Channel can access the VME Master Interface.

## 2.2.1.2 Request Modes

### Request Levels

The Universe is software configurable to request on all VMEbus request levels: BR3\*, BR2\*, BR1\*, and BR0\*. The default setting is for level 3 VMEbus request. The request level is a global programming option set through the VRL bits in the MAST\_CTL register (Table A.56). The programmed request level is used by the VME Master Interface regardless of the channel (Interrupt Channel, DMA Channel, or PCI Slave Channel) currently accessing the VME Master Interface.

### Fair and Demand

The Universe requester may be programmed for either Fair or Demand mode. The request mode is a global programming option set through the VRM bits in the MAST\_CTL register (Table A.56).

In Fair mode, the Universe does not request the VMEbus until there are no other VMEbus requests pending at its programmed level. This mode ensures that every requester on a given level has access to the bus.

In Demand mode (the default setting), the requester asserts its bus request regardless of the state of the BRn\* line. By requesting the bus frequently, requesters far down the daisy chain may be prevented from ever obtaining bus ownership. This is referred to as “starving” those requesters. Note that in order to achieve fairness, all bus requesters in a VME system must be set to Fair mode.

### 2.2.1.3 VMEbus Release

The Universe VMEbus requester can be configured as either RWD (release when done) or ROR (release on request) using the VREL bit in the MAST\_CTL register (Table A.56). The default setting is for RWD. ROR means the Universe releases BBSY\* only if a bus request is pending from another VMEbus master and once the channel that is current owner of the VME Master Interface is done. Ownership of the bus may be assumed by another channel without re-arbitration on the bus if there are no pending requests on any level on the VMEbus. When set for RWD, the VME Master Interface releases BBSY\* when the channel accessing the VME Master Interface is done (see below). Note that the MYBBSY status bit in the MISC\_STAT register (Table A.58) is set while the Universe asserts the BBSY\* output.

The VMEbus is released (in RWD mode) when the channel (for example, the DMA Channel) is done, even if another channel has a request pending (for example, the PCI Slave Channel). A re-arbitration of the VMEbus is required for any pending channel requests. Each channel has a set of rules that determine when it is 'done' with its VMEbus transaction.

The Interrupt Channel is done when a single interrupt acknowledge cycle is complete.

The PCI Slave Channel is done under the following conditions:

- when the TXFIFO is empty (the TXFE bit is set in the MISC\_STAT register, Table A.58),
- when the maximum number of bytes per PCI Slave Channel tenure has been reached (as programmed with the PWON field in the MAST\_CTL register, Table A.56),
- coupled cycle is complete and the Coupled Window Timer has expired,
- the Coupled Request Timer expires before a coupled cycle is retried by a PCI master, or
- when VMEbus ownership is acquired with the VOWN bit and then the VMEbus ownership bit cleared (see "Exclusive Accesses" on page 2-41).

The DMA Channel is done under the following conditions (see "FIFO Operation and Bus Ownership" on page 2-82 and "DMA Error Handling" on page 2-86):

- DMAFIFO full during VMEbus to PCI bus transfers,
- DMAFIFO empty during PCI bus to VMEbus transfers,
- if an error is encountered during the DMA operation,
- the DMA VMEbus Tenure Byte Counter has expired, or
- DMA block is complete.

The Universe does not monitor BCLR\* and so its ownership of the VMEbus is not affected by the assertion of BCLR\*.

## 2.2.2 Universe as VME Master

The Universe becomes VMEbus master as a result of the following chain of events:

1. a PCI master accesses a Universe PCI slave image (leading to VMEbus access) or the DMA Channel initiates a transaction,
2. either the Universe PCI Slave Channel or the DMA Channel wins access to the VME Master Interface through internal arbitration, and
3. the Universe Master Interface requests and obtains ownership of the VMEbus.

The Universe will also become VMEbus master if the VMEbus ownership bit is set (see “Exclusive Accesses” on page 2-41) and in its role in VMEbus interrupt handling (see “VMEbus Interrupt Handling” on page 2-61).

The following sections describe the function of the Universe as a VMEbus master in terms of the different phases of a VMEbus transaction: addressing, data transfer, cycle termination, and bus release.

### 2.2.2.1 Addressing Capabilities

Depending upon the programming of the PCI slave image (see “PCI Bus Slave Images” on page 2-47), the Universe generates A16, A24, A32, and CR/CSR address phases on the VMEbus. The address mode and type (supervisor/non-privileged and program/data) are also programmed through the PCI slave image. Address pipelining is provided except during MBLT cycles, where the VMEbus specification does not permit it.

The address and AM codes that are generated by the Universe are functions of the PCI address and PCI slave image programming (see “PCI Bus Slave Images” on page 2-47) or through DMA programming.

The Universe generates Address-Only-with-Handshake (ADOH) cycles in support of lock commands for A16, A24, and A32 spaces. ADOH cycles must be generated through the Special Cycle Generator (see “RMW and ADOH Cycles” on page 2-40).

To increase the flexibility of the Universe’s address space programming, there are two User Defined AM codes that can be programmed through the USER\_AM register (Table A.59). After power-up, the two values in the USER\_AM register default to the same VME64 user-defined AM code.

### 2.2.2.2 Data Transfer Capabilities

The data transfer between the PCI bus and VMEbus is perhaps best explained by Figure 2.2 on page 2-11. The Universe can be seen as a funnel where the mouth of the funnel is the data width of the PCI transaction. The end of the funnel is the maximum VMEbus data width programmed into the PCI slave image. For example, consider a 32bit PCI transaction accessing a PCI slave image with VDW set to 16 bits. A data beat with all byte lanes enabled will be broken into two 16-bit cycles on the VMEbus. If the PCI slave image is also programmed with block transfers enabled, the 32-bit PCI data beat will result in a D16 block transfer on the VMEbus. Write data is unpacked to the VMEbus and read data is packed to the PCI bus data width.

If the data width of the PCI data beat is the same as the maximum data width of the PCI slave image, then the Universe maps the data beat to an equivalent VMEbus cycle. For example, consider a 32bit PCI transaction accessing a PCI slave image with VDW set to 32 bits. A data beat with all byte lanes enabled is translated to a single 32-bit cycle on the VMEbus.

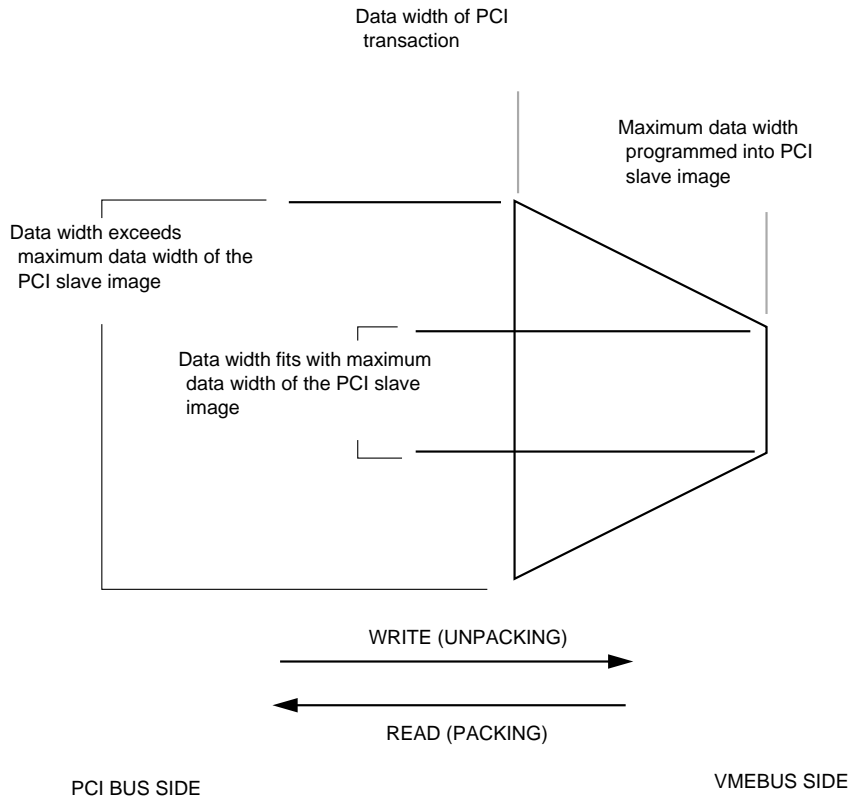
As the general rule, if the PCI bus data width is less than the VMEbus data width then there is no packing or unpacking between the two buses. The only exception to this is during 32-bit PCI multi-data beat transactions to a PCI slave image programmed with maximum VMEbus data width of 64 bits. In this case, packing/unpacking occurs to make maximum use of the full bandwidth on both buses.

Only aligned VMEbus transactions are generated, so if the requested PCI data beat has unaligned or non-contiguous byte enables, then it is broken into multiple aligned VMEbus transactions no wider than the programmed VMEbus data width. For example, consider a three-byte PCI data beat (on a 32-bit PCI bus) accessing a PCI slave image with VDW set to 16 bits. The three-byte PCI data beat will be broken into two aligned VMEbus cycles: a single-byte cycle and a double-byte cycle (the ordering of the two cycles depends on the arrangement of the byte enables in the PCI data beat). If in the above example the PCI slave image has a VDW set to 8 bits, then the three-byte PCI data beat will be broken into three single-byte VMEbus cycles.

BLT/MBLT cycles are initiated on the VMEbus if the PCI slave image has been programmed with this capacity (see “PCI Bus Slave Images” on page 2-47). The length of the BLT/MBLT transactions on the VMEbus will be determined by the initiating PCI transaction or the setting of the PWON field in the MAST\_CTL register (Table A.56). For example, a single data beat PCI transaction queued in the TXFIFO results in a single data beat block transfer on the VMEbus. With the PWON field, the user can specify a transfer byte count that will be dequeued from the TXFIFO before the VME Master Interface relinquishes the VMEbus.

During DMA operations, the Universe will attempt block transfers to the maximum length permitted by the VMEbus specification (256 bytes for BLT, 2 Kbytes for MBLT) and as limited by the VON counter.





**Figure 2.2 : Influence of Transaction Data Width and Slave Image Data Width on Data Packing/Unpacking**

The Universe provides indivisible transactions with the VMEbus lock commands and the VMEbus ownership bit (see “Exclusive Accesses” on page 2-41).

### 2.2.2.3 Cycle Terminations

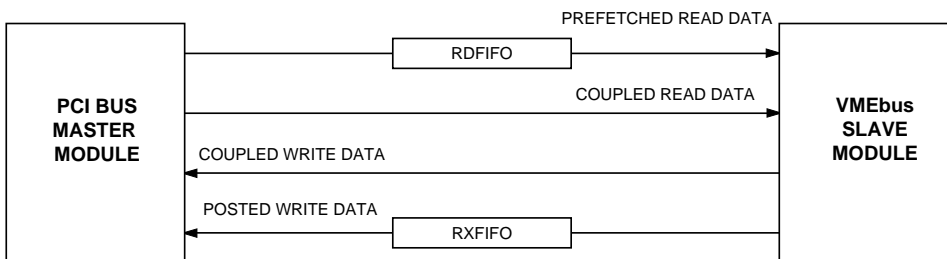
The Universe accepts BERR\* or DTACK\* as cycle terminations from the VMEbus slave. The assertion of BERR\* indicates that some type of system error occurred and the transaction did not complete properly. A VMEbus BERR\* received by the Universe during a coupled transaction is communicated to the PCI master as a target abort. No information is logged if the Universe receives BERR\* in a coupled transaction. If an error occurs during a posted write to the VMEbus, the Universe uses the V\_AMERR register (Table A.80) to log the AM code of the transaction (AMERR [5:0]), and the state of the IACK\* signal (IACK bit, to indicate whether the error occurred during an IACK cycle). The current transaction in the FIFO is purged. The V\_AMERR register also records if multiple errors have occurred (with the M\_ERR bit), although the actual number of errors is not given. The error log is qualified by the value of the V\_STAT bit. The address of the errored transaction is latched in the V\_AERR register (Table A.81). When the Universe receives a VMEbus error during a posted write, it generates an interrupt on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “The Interrupt Channel” on page 2-56, Table A.43 and Table A.44).

DTACK\* signals the successful completion of the transaction.

### 2.2.3 Universe as VME Slave

The Universe becomes VMEbus slave when one of its four programmed slave images or register images are accessed by a VMEbus master (note that the Universe cannot reflect a cycle on the VMEbus and access itself). Depending upon the programming of the slave image, different possible transaction types result (see “VME Slave Images” on page 2-44 for a description of the types of accesses to which the Universe responds).

For reads, the transaction can be coupled or prefetched. Similarly, write transactions can be coupled or posted. The type of read or write transaction allowed by the slave image is dependent upon the programming of that particular VME slave image (see Figure 2.3 below and “VME Slave Images” on page 2-44). To ensure sequential consistency, prefetched reads, coupled reads, and coupled write operations are only processed once all previously posted write operations have completed (i.e. the RXFIFO is empty).



**Figure 2.3 : VMEbus Slave Channel Dataflow**

Incoming cycles from the VMEbus can have data widths of 8-bits, 16-bits, 32-bits, and 64-bits. Although the PCI bus supports only two port sizes (32-bits and 64-bits), the byte lanes on the PCI bus can be individually enabled, which allows each type of VMEbus transaction to be directly mapped to the PCI data bus.



*In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).*

### 2.2.3.1 Coupled Transfers

A coupled transfer means that no FIFO is involved in the transaction and handshakes are relayed directly through the Universe. Coupled mode is the default setting for the VMEbus slave images. Coupled transfers only proceed once all posted write entries in the RXFIFO have completed (see “Posted Writes” below).

A coupled cycle with multiple data beats (i.e. block transfers) on the VMEbus side is always mapped to single data beat transactions on the PCI bus, where each data beat on the VMEbus is mapped to a single data beat transaction on the PCI bus regardless of data beat size. No packing or unpacking is performed. The only exception to this is when a D64 VMEbus transaction is mapped to D32 on the PCI bus. The data width of the PCI bus is dependent upon the programming of the VMEbus slave image (32-bit or 64-bit, see “VME Slave Images” on

page 2-44). The Universe enables the appropriate byte lanes on the PCI bus as required by the VMEbus transaction. For example, a VMEbus slave image programmed to generate 32-bit transactions on the PCI bus is accessed by a VMEbus D08 BLT read transaction (prefetching is not enabled in this slave image). The transaction is mapped to single data beat 32-bit transfers on the PCI bus with only one byte lane enabled.

The Universe does not generate RETRY\* on the VMEbus, so a target-retry from a PCI slave will not be communicated to the VMEbus master. PCI transactions terminated with target abort or master abort are terminated on the VMEbus with BERR\*. Note that the Universe sets the R\_TA or R\_MA bits in the PCI\_CS register (Table A.3) when it receives a target abort or master abort.

### 2.2.3.2 Posted Writes

A posted write involves the VMEbus master writing data into the Universe's RXFIFO, rather than directly to the PCI address. Write transactions from the VMEbus are processed as posted if the PWEN bit is set in the VMEbus slave image control register (see "VME Slave Images" on page 2-44). If the bit is cleared (the default setting) the transaction bypasses the FIFO and is performed as a coupled transfer (see above). Incoming posted writes from the VMEbus are queued in a 16-entry deep RXFIFO. Each entry in the RXFIFO can contain 64 address bits (with extra bits provided in an address entry for command information), or 64 data bits. Each incoming VMEbus address phase, whether it is 16-bit, 24-bit, or 32-bit, constitutes a single entry in the RXFIFO and is followed by subsequent data entries. The address entry contains the translated PCI address space and command information mapping relevant to the particular VMEbus slave image that has been accessed (see "VME Slave Images" on page 2-44). For this reason, any re-programming of VMEbus slave image attributes will only be reflected in RXFIFO entries queued after the re-programming. Transactions queued before the re-programming are delivered to the PCI bus with the VMEbus slave image attributes that were in use before the re-programming.

Incoming non-block write transactions from the VMEbus require two entries in the RXFIFO: one address entry (with accompanying command information) and one data entry. The size of the data entry corresponds to the data width of the VMEbus transfer. Block transfers require at least two entries: one entry for address and command information, and one or more data entries. The VME Slave Channel packs data received during block transfers to the full 64-bit width of the RXFIFO. For example, a ten data phase D16 BLT transfer (20 bytes in total) does not require ten data entries in the RXFIFO. Instead, eight of the ten data phases (16 bits per data phase for a total of 128 bits) are packed into two 64-bit data entries in the RXFIFO. The final two data phases (32 bits combined) are queued in the next RXFIFO entry. When you add the address entry to the three data entries, this VMEbus block write has been stored in a total of four RXFIFO entries.

Unlike the PCI Slave Channel (see page 2-33), the VME Slave Channel does not retry the VMEbus if the RXFIFO does not have enough space to hold an incoming VMEbus write transaction. Instead, the DTACK\* response from the VME Slave Interface is delayed until space becomes available in the RXFIFO. Since single transfers require two entries in the RXFIFO, two entries must be freed up before the VME Slave Interface asserts DTACK\*. Similarly, the VME Slave Channel requires two available RXFIFO entries before it can acknowledge the first data phase of a BLT or MBLT transfer (one entry for the address phase and one for the first data phase). If the RXFIFO has no available space for subsequent data phases in the block transfer, then the VME Slave Interface delays assertion of DTACK\* until a single entry is available for the next data phase in the block transfer.

The availability of RXFIFO entries depends upon how many transactions are queued in the RXFIFO. The VME Slave Channel permits only four transactions queued in the RXFIFO at any one time. A transaction is a VMEbus transaction with an address phase and one or more data phases. For example a single word write is one transaction. If four single word writes are queued in the RXFIFO, then the RXFIFO is considered full. If four transactions are already queued in the RXFIFO, assertion of DTACK\* is delayed until one transaction is delivered to the PCI bus.

The PCI Master Interface uses transactions queued in the RXFIFO to generate transactions on the PCI bus. No address phase deletion is performed, so the length of a transaction on the PCI bus corresponds to the length of the queued VMEbus transaction. Non-block transfers are generated on the PCI bus as single data beat transactions. Block transfers are generated as one or more burst transactions, where the length of the burst transaction is programmed by the PABS bit in the MAST\_CTL register.

The Universe always packs or unpacks data from the VMEbus transaction to the PCI bus data width programmed into the VMEbus slave image (with all PCI bus byte lanes enabled). For example, consider a VMEbus slave image programmed for posted writes and a D32 PCI bus that is accessed with a VMEbus D16 block write transaction. The VMEbus D16 write transaction is mapped to a D32 write transaction on the PCI bus with all byte lanes enabled. (However, note that a single D16 transaction from the VMEbus is mapped to the PCI bus as D32 with only two byte lanes enabled).

During block transfers, the Universe will pack data to the full negotiated width of the PCI bus. This may imply that for block transfers that begin or end on addresses not aligned to the PCI bus width different byte lanes may be enabled during each data beat.

If an error occurs during a posted write to the PCI bus, the Universe uses the L\_CMDERR register (Table A.31) to log the command information for the transaction (CMDERR [3:0]). The L\_CMDERR register also records if multiple errors have occurred (with the M\_ERR bit) although the actual number is not given. The error log is qualified with the L\_STAT bit. The address of the errored transaction is latched in the LAERR register (Table A.32). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “Bus Error Handling” on page 2-52 and “The Interrupt Channel” on page 2-56).

### 2.2.3.3 Prefetched Block Reads

Prefetching of read data occurs for VMEbus block transfers (BLT, MBLT) in those slave images that have the prefetch enable (PREN) bit set (see “VME Slave Images” on page 2-44). Without prefetching, block read transactions from a VMEbus master are handled by the VME Slave Channel as coupled reads. This means that each data phase of the block transfer is translated to a single data beat transaction on the PCI bus. In addition, only the amount of data requested during the relevant data phase is fetched from the PCI bus. For example, D16 block read transaction with 32 data phases on the VMEbus maps to 32 PCI bus transactions, where each PCI bus transaction has only two byte lanes enabled. Note the VMEbus lies idle during the arbitration time required for each PCI bus transaction, resulting in a considerable performance degradation.

With prefetching enabled, the VME Slave Channel uses a 16 entry deep RDFIFO to provide read data to the VMEbus with minimum latency. The RDFIFO is 64 bits wide, with additional bits for control information. If a VMEbus slave image is programmed for prefetching (see “VME Slave Images” on page 2-44), then a block read access to that image causes the VME Slave Channel to generate aligned burst read transactions on the PCI bus (the size of the burst read transactions is determined by the setting of the aligned burst size, PABS in the MAST\_CTL register). These PCI burst read transaction are queued in the RDFIFO and the data is then delivered to the VMEbus. Note that the first data phase provided to the VMEbus master is essentially a coupled read, but subsequent data phases in the VMEbus block read are delivered from the RDFIFO and are essentially decoupled (see “Prefetched Reads” on page 2-54 for the impact on bus error handling).

The data width of the transaction on the PCI bus (32-bit or 64-bit) depends upon the setting of the LD64EN bit in the VME slave image control register (e.g. see Table A.60) and the capabilities of the accessed PCI slave. Internally, the prefetched read data is packed to 64 bits, regardless of the width of the PCI bus or the data width of the original VMEbus block read (no address information is stored with the data). Once one entry is queued in the RDFIFO, the VME Slave Interface delivers the data to the VMEbus, unpacking the data as necessary to fit with the data width of the original VMEbus block read (e.g. D16, or D32). The VME Slave Interface continuously delivers data from the RDFIFO to the VMEbus master performing the block read transaction. Because PCI bus data transfer rates exceed those of the VMEbus, it is unlikely that the RDFIFO will ever be unable to deliver data to the VMEbus master. For this

reason, block read performance on the VMEbus will be similar to that observed with block writes. However, should the RDFIFO be unable to deliver data to the VMEbus master (which may happen if there is considerable traffic on the PCI bus or the PCI bus slave has a slow response) the VME Slave Interface delays DTACK\* assertion until an entry is queued and is available for the VMEbus block read.

On the PCI side, prefetching continues as long as there is room for another transaction in the RDFIFO and the initiating VMEbus block read is still active. The space required in the RDFIFO for another PCI burst read transaction is determined by the setting of the PCI aligned burst size (PABS in the MAST\_CTL register, Table A.56). If PABS is set for 32 bytes, there must be four entries available in the RDFIFO; for aligned burst size set to 64 bytes, eight entries must be available. When there is insufficient room in the RDFIFO to hold another PCI burst read, the read transactions on the PCI bus are terminated and only resume if room becomes available for another aligned burst AND the original VMEbus block read is still active. When the VMEbus block transfer terminates, any remaining data in the RDFIFO is purged.

Regardless of the read request, the data width of prefetching on the PCI side is full width with all byte lanes enabled. If the request is unaligned, then the first PCI data beat will have only the relevant byte lanes enabled. Subsequent data beats will have full data width with all byte lanes enabled. If LD64EN is set in the VME Slave image, the Universe requests D64 on the PCI bus by asserting REQ64# during the address phase. If the PCI slave does not respond with ACK64#, subsequent data beats are D32.

The Universe only translates errors from the PCI bus to the VMEbus during the first data beat (the coupled portion) of the prefetched read. A target abort on the PCI bus is translated as a BERR\* signal on the VMEbus. However, a target abort on subsequent data beats during a prefetched read is ignored, but does cause prefetching to stop. There is no logging of the error, and all previously fetched data is provided to the VMEbus master. Once the block read on the VMEbus extends beyond the available data in the RDFIFO, then a new prefetch is initiated on the PCI bus. The point at which the new prefetch is initiated corresponds to where the error previously occurred. If the error occurs again, it will be on the first (coupled) data beat of the transaction and will be translated to the VMEbus as a BERR\* signal. As with all coupled errors, no error is logged and no interrupt is generated. It should be anticipated that two target aborts may be generated before the VMEbus block read is terminated with BERR\*.

### 2.2.3.4 VMEbus Lock Commands

The Universe supports VMEbus lock commands as described in the VME64 specification. Under the specification, ADOH cycles are used to execute the lock command (with a special AM code). Any resource locked on the VMEbus cannot be accessed by any other resource during the bus tenure of the VMEbus master. If the Universe receives a VMEbus lock command, it asserts LOCK# to the addressed resource on the PCI bus. The Universe holds the PCI bus lock until the VMEbus lock command is terminated, i.e. when BBSY\* is negated. All subsequent slave VMEbus transactions are coupled while the Universe owns PCI LOCK#. Note that the VME Slave Channel has dedicated access to the PCI Master Interface during the locked transaction.

### 2.2.3.5 VMEbus Read Modify Writes

A read-modify-write (RMW) cycle allows a VMEbus master to read from a VMEbus slave and then write to the same resource without relinquishing bus tenure between the two operations. Each of the Universe slave images can be programmed to map RMW transactions to PCI locked transactions. If the LLRMW enable bit is set in the appropriate VMEbus slave image control register (e.g. Table A.60), then every non-block slave read is mapped to a coupled PCI locked read. LOCK# will be held on the PCI bus until AS\* is negated on the VMEbus. Every non-block slave read is assumed to be a RMW since there is no possible indication from the VMEbus master that the single cycle read is just a read or the beginning of a RMW.

If the LLRMW enable bit is not set and the Universe receives a VME RMW cycle, the read and write portions of the cycle will be treated as independent transactions on the PCI bus: i.e., a read followed by a write. The write may be coupled or decoupled depending on the state of the PWEN bit in the accessed slave image.



*Note: There may be a performance loss for reads that are processed through a RMW-capable slave image. Some of this comes about due to the sampling of and arbitration for LOCK# by the Universe's PCI Master Interface. More of a performance loss can arise if LOCK# is currently owned by another PCI master.*

### 2.2.3.6 Register Accesses

See “Registers” on page 2-87 for a full description of register mapping and register access.



### 2.2.3.7 DTACK\* Rescinding

The user can increase performance in many systems by programming the Universe to rescind DTACK\*. When the Universe releases the VMEbus, it can be programmed to either just release DTACK\* (in which case DTACK\* floats to its negated level) or rescind DTACK\* (in which case the Universe actively drives DTACK\* to its negated level). DTACK\* rescinding is programmed by setting the RESCIND bit in the MISC\_CTL register (Table A.57) and is enabled by default.

## 2.2.4 VMEbus Configuration

The Universe provides the following functions to assist in the initial configuration of the VMEbus system:

- First Slot Detector,
- Register Access at Power-up, and
- Auto Slot ID (two methods).

These are described separately below.

### 2.2.4.1 First Slot Detector

As specified by the VME64 specification the First Slot Detector module on the Universe samples BG3IN\* immediately after reset to determine whether the Universe's host board resides in slot 1. The VMEbus specification requires that BG[3:0]\* lines be driven high after reset. This means that if a card is preceded by another card in the VMEbus system, it will always sample BG3IN\* high after reset. BG3IN\* can only be sampled low after reset by the first card in the system (there is no preceding card to drive BG3IN\* high). If BG3IN\* is sampled at logic low immediately after reset (due to the Universe's internal pull-down), then the Universe's host board is in slot 1 and the Universe becomes SYSCON; otherwise, the SYSCON module is disabled. This mechanism may be overridden by software through clearing or setting the SYSCON bit in the MISC\_CTL register (Table A.57).

### 2.2.4.2 VMEbus Register Access at Power-up

The Universe provides a VMEbus slave image that allows access to all Universe Control and Status Registers (UCSR). The base address for this slave image is programmed through the VRAI\_BS register (Table A.77). At power-up, the Universe can program the VRAI\_BS and VRAI\_CTL (Table A.76) registers with information specifying the UCSR slave image (see "Power-Up Options" on page 2-98).

Register access at power-up would be used in systems where the Universe's card has no CPU, or where register access for that card needs to be independent of the local CPU.

## 2.2.5 Automatic Slot Identification

The Universe supports two types of Auto-ID functionality. One type uses the Auto Slot ID technique as described in the VME64 specification. The other type uses a proprietary method developed by DY4 Systems and implemented in the Tundra SCV64. Neither system identifies geographical addressing, only the relative position amongst the boards present in the system (i.e. fourth board versus fourth slot).

Auto-ID prevents the need for jumpers to uniquely identify cards in a system. This can:

- increase the speed of system level repairs in the field,
- reduce the possibility of incorrect configurations, and
- reduce the number of unique spare cards that must be stocked.

Both methods of Auto ID employed by the Universe are described below.

### 2.2.5.1 Auto Slot ID: VME64 Specified

The VME64 auto ID cycle (described in the VME64 Specification) requires at power-up that the Auto ID slave

- generate IRQ2\*, and
- negate SYSFAIL\*.

When the Auto ID slave responds to the Monarch's IACK cycle, it will

- enable accesses to its CR/CSR space,
- provide a Status/ID to the Monarch indicating the interrupt is an Auto-ID request,
- assert DTACK\*, and
- release IRQ2\*.

The Universe participates in the VME64 auto ID cycle in either an automatic or semi-automatic mode. In its fully automatic mode, it holds SYSFAIL\* asserted until SYSRST\* is negated. When SYSRST\* is negated, the Universe asserts IRQ2\* and releases SYSFAIL\*. In its semi-automatic mode, the Universe still holds SYSFAIL\* asserted until SYSRST\* is negated. However, when SYSRST\* is negated, the local CPU performs diagnostics and local logic sets the AUTOID bit in the MISC\_CTL register (Table A.57). This asserts IRQ2\* and releases SYSFAIL\*.

After SYSFAIL\* is released and the Universe detects a level 2 IACK cycle, it responds with the STATUS/ID stored in its level 2 STATID register (which defaults to 0xFE).

The Universe can be programmed so that it will not release SYSFAIL\* until the SYSFAIL bit in the VCSR\_CLR register (Table A.82) is cleared by local logic (SYSFAIL\* is asserted if the SYSFAIL bit in the VCSR\_SET register, Table A.83, is set at power-up). Since the system Monarch does not service the Auto-ID slave until after SYSFAIL\* is negated, not clearing the SYSFAIL bit allows the Auto-ID process to be delayed until the CPU completes local diagnostics. Once local diagnostics are complete, the CPU clears the SYSFAIL bit and the Auto-ID cycle proceeds.

The Monarch can perform CR/CSR reads and writes at A[23:19]0x00 in CR/CSR space and re-locate the Universe's CR/CSR base address.

### Universe and the Auto-ID Monarch

At power-up an Auto-ID Monarch waits to run a level 2 IACK cycle until after SYSFAIL\* goes high. After the IACK cycle is performed and it has received a Status/ID indicating an Auto-ID request, the monarch software

- masks IRQ2\* (so that it will not service other interrupters at that level until current Auto-ID cycle is completed),
- performs an access at 0x00 in CR/CSR space to get information about Auto-ID slave,
- moves the CR/CSR base address to a new location, and
- unmask IRQ2\* (to allow it to service the next Auto-ID slave).

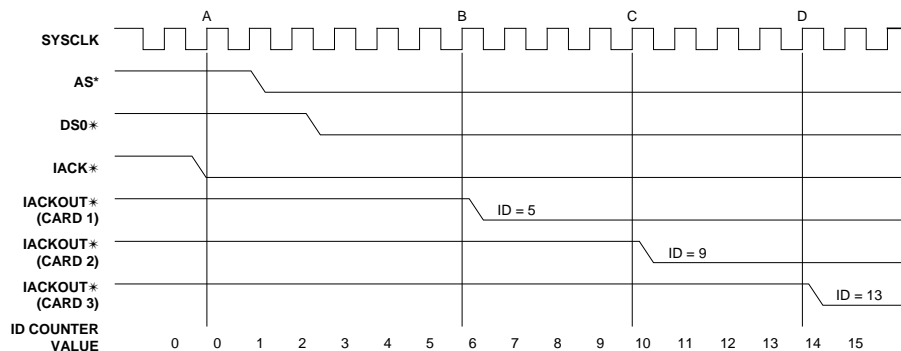
The Universe supports monarch activity through its capability to be a level 2 interrupt handler. All other activity must be handled through software residing on the board.

### 2.2.5.2 Auto-ID: A Proprietary Tundra Method

The Universe uses a proprietary Auto-ID scheme if this is selected as a power-up option (see "Auto-ID" on page 2-101). The Tundra proprietary Auto-ID function identifies the relative position of each board in the system, without using jumpers or on-board information. The ID number generated by Auto-ID can then be used to determine the board's base address.

After any system reset (assertion of SYSRST\*), the Auto-ID logic responds to the first level one IACK cycle on the VMEbus.

After the level one IACK\* signal has been asserted (either through IRQ1\* or with a synthesized version), the Universe in slot 1 counts five clocks from the start of the cycle and then asserts IACKOUT\* to the second board in the system (see Figure 2.4). All other boards continue counting until they receive IACKIN\*, then count four more clocks and assert IACKOUT\* to the next board. Finally, the last board asserts IACKOUT\* and the bus pauses until the data transfer time-out circuit ends the bus cycle by asserting BERR\*.



**Figure 2.4 : Timing for Auto-ID Cycle**

Because all boards are four clocks “wide”, the value in the clock counter is divided by four to identify the slot in which the board is installed; any remainder is discarded. Note that since the start of the IACK cycle is not synchronized to SYSCLK, a one count variation from the theoretical value of the board can occur. However, in all cases the ID value of a board is greater than that of a board in a lower slot number. The result is placed in the DY4AUTOID [7:0] field and the DY4DONE bit is set (both are located in the MISC\_STAT register, Table A.58).

## 2.2.6 System Controller Functions

When located in Slot 1 of the VME system (see “First Slot Detector” on page 2-19), the Universe assumes the role of SYSCON and sets the SYSCON status bit in the MISC\_CTL register (Table A.57). In accordance with the VME64 specification, as SYSCON the Universe provides:

- a system clock driver,
- an arbitration module,
- an IACK Daisy Chain Driver (DCD), and
- a bus timer.

### 2.2.6.1 System Clock Driver

The Universe provides a 16 MHz SYSCLK signal derived from CLK64 when configured as SYSCON.

### 2.2.6.2 VMEbus Arbiter

When the Universe is SYSCON, the Arbitration Module is enabled. The Arbitration Module supports the following arbitration modes:

- Fixed Priority Arbitration Mode (PRI),
- Single Level Arbitration (SGL) (a subset of PRI), or
- Round Robin Arbitration Mode (RRS) (default setting).

These are set with the VARB bit in the MISC\_CTL register (Table A.57).

#### Fixed Priority Arbitration Mode (PRI)

In this mode, the order of priority is VRBR3#, VRBR2#, VRBR1#, and VRBR0# as defined by the VME64 specification. The Arbitration Module issues a Bus Grant (VBGO [3:0]#) to the highest requesting level.

If a Bus Request of higher priority than the current bus owner becomes asserted, the Arbitration Module asserts VBCLR# until the owner releases the bus (VRBBSY# is negated).

#### Single Level Arbitration Mode (SGL)

In this mode, a subset of priority mode, all requests and grants are made exclusively on level 3. Set the Universe in PRI mode to use this mode.

#### Round Robin Arbitration Mode (RRS)

This mode arbitrates all levels in a round robin mode, repeatedly scanning from levels 3 to 0. Only one grant is issued per level and one owner is never forced from the bus in favor of another requester (VBCLR# is never asserted).

Since only one grant is issued per level on each round robin cycle, several scans will be required to service a queue of requests at one level.

#### VMEbus Arbiter Time-out

The Universe's VMEbus arbiter can be programmed to time-out if the requester does not assert BBSY\* within a specified period. This allows BGOUT to be negated so that the arbiter may continue with other requesters. The timer is programmed using the VARBTO field in the MISC\_CTL register (Table A.57), and can be set to 16  $\mu$ s, 256  $\mu$ s, or disabled. The default setting for the timer is 16  $\mu$ s.

### 2.2.6.3 IACK Daisy-Chain Driver Module

The IACK Daisy-Chain Driver module is enabled when the Universe becomes system controller. This module guarantees that IACKIN\* will stay high for at least 30ns as specified in rule 40 of the VME64 specification.

### 2.2.6.4 VMEbus Time-out

A programmable bus timer allows users to select a VMEbus time-out period. The time-out period is programmed through the VBTO field in the MISC\_CTL register (Table A.57) and can be set to 16 $\mu$ s, 32 $\mu$ s, 64 $\mu$ s, 128  $\mu$ s, 256  $\mu$ s, 512  $\mu$ s, 1024  $\mu$ s, or disabled. The default setting for the timer is 64  $\mu$ s. The VMEbus Timer module asserts VXBERR# if a VMEbus transaction times out (indicated by one of the VMEbus data strobes remaining asserted beyond the time-out period).

### 2.2.7 BI-Mode<sup>®</sup>

BI-Mode (Bus Isolation Mode) is a mechanism for logically isolating the Universe from the VMEbus. Entering BI-mode disables communication between the PCI bus and VMEbus, pointing all transceivers inwards. Interrupt and arbitration daisy chains as well as all SYSCON functions remain operative, but the Universe ceases all VMEbus master and most slave activity. The VMEbus slave images are disabled, but the CR/CSR and UCSRA images are accessible. A PCI master attempting to access a VMEbus resource will be retried. By isolating boards from the VMEbus, the user can implement:

- hot-standby systems
- system diagnostics for routine maintenance, or
- fault isolation in the event of a card failure.

There are three ways to cause the Universe to enter BI-Mode. The Universe is put into BI-Mode if:

1. the BI-mode power-up option is selected (See “Power-up Option Descriptions” on page 2-99 and Table 2.21 on page 2-102). If the Universe is powered-up in BI-mode, then (a) the BI bit in the MISC\_CTL register (Table A.57) is set, and (b) any future SYSRST\* or PWRRST# will restore the Universe to BI-Mode,
2. the BI bit in the MISC\_CTL register is set, or
3. VRIRQ# [1] is asserted, provided that the ENGBI bit in the MISC\_CTL register has been set.

The only way to remove the Universe from BI-mode is to clear the BI bit in the MISC\_CTL register, provided that the source of the BI-mode is no longer active (i.e., if VRIRQ# [1] is still being asserted while the ENGBI bit in the MISC\_CTL register is set, then attempting to clear the BI bit in the MISC\_CTL register will not be effective.) Furthermore, as mentioned above, if the Universe has been powered-up in BI-mode, any future SYSRST\* or PWRRST# will restore the Universe to BI-Mode.

## 2.3 PCI Bus Interface

The PCI Bus Interface is organized as follows:

- “PCI Cycles - Overview” below,
- “Universe as PCI Master” on page 2-30, and
- “Universe as PCI Slave” on page 2-33.

The Universe PCI bus interface is electrically and logically directly connected to the PCI bus. For information concerning the different types of PCI accesses available, see “PCI Bus Slave Images” on page 2-47.

### 2.3.1 PCI Cycles - Overview

The PCI bus port of the Universe operates as a PCI compliant port with a 64-bit multiplexed address/data bus. The Universe PCI bus interface is configured as little-endian using address invariant translation when mapping between the VMEbus and the PCI bus. Address invariant translation preserves the byte ordering of a data structure in a little-endian memory map and a big-endian memory map (see Appendix-E).

The Universe has all the PCI signals described in the PCI specification with the exception of SBO# and SDONE (since the Universe does not provide cache support).

Universe PCI cycles are synchronous, meaning that bus and control input signals are externally synchronized to the PCI clock (CLK). PCI cycles are divided into four phases:

1. request,
2. address phase,
3. data transfer, and
4. cycle termination.

#### 2.3.1.1 32-Bit Versus 64-Bit PCI

At reset, the Universe is configured with a 32-bit or 64-bit PCI data bus, depending on the level of REQ64#. The Universe is configured as 32-bit PCI if REQ64# is high; it is configured as 64-bit if REQ64# is low. Accordingly, users must ensure that REQ64# is driven high or low at reset; otherwise, REQ64# will be floating and whether the Universe is reset as a 64-bit or 32-bit PCI is indeterminate. In a 64-bit PCI bus, an external resource drives REQ64# low; this enables the Universe as 64-bit. To configure the device for a 32-bit PCI bus, pull down

REQ64# with a 4.7k $\Omega$  resistor. On a 32-bit PCI bus, the Universe drives all its 64-bit extension bi-direct signals at all times; these signals include: C/BE[7:4]#, AD[63:32], REQ64#, PAR64# and ACK64# to unknown values. Independent of the setting of the LD64EN bit, the Universe will never attempt a 64-bit cycle on the PCI bus if it is powered up as 32-bit.

Each of the Universe's VMEbus slave images can be programmed so that VMEbus transactions are mapped to a 64-bit data bus on the PCI interface (with the LD64EN bit, e.g. see Table A.60). If the VMEbus slave image is programmed with a 64-bit PCI bus data width and if the Universe powered up in a 64-bit PCI environment, then the Universe asserts REQ64# during the address phase of the PCI transaction. If the PCI slave is 64-bit capable, then it will respond with ACK64# and the Universe will pack data to the full width (64 bits) of the PCI bus. If the PCI slave is not 64-bit capable, then it does not assert ACK64# and the Universe will pack data to a 32-bit PCI bus.



*Note that REQ64# will be asserted if LD64EN is set in a 64-bit PCI system independent of whether the Universe has a full 64 bits to transfer. This may result in a performance degradation because of the extra clocks required to assert REQ64# and to sample ACK64#. Also, there can be some performance degradation when accessing 32-bit targets with LD64EN set. Do not set this bit unless there are 64-bit targets in the slave image window.*

If the Universe VME slave images are not programmed for a 64-bit wide PCI data bus, then the Universe operates transparently in a 32-bit PCI environment.

### 2.3.1.2 PCI Bus Request

When the Universe requires control of the PCI bus it asserts REQ# and gains bus mastership when the PCI arbiter asserts GNT#. GNT# assertion allows bus access for a single transaction. If the arbiter removes GNT# after the Universe has begun its PCI transaction, the Universe completes the current transaction and releases the PCI bus (removes REQ#). This means that the Universe PCI Master Interface may have to re-arbitrate for the PCI bus after every transaction if its GNT# is removed.

### 2.3.1.3 Address Phase

PCI transactions are initiated by asserting FRAME# and driving address and command information onto the bus. In the VMEbus Slave Channel, the Universe calculates the address for the PCI transaction by adding a translation offset to the VMEbus address (see "Universe as VME Slave" on page 2-12).

The command signals (on the C/BE# lines) contain information about memory space, cycle type and whether the transaction is read or write. Table 2.2 below gives PCI the command type encoding implemented with the Universe.



**Table 2.2 : Command Type Encoding for Transfer Type**

<b>C/BE# [3:0] for PCI, C/BE# [7:4] for non-multiplexed</b>	<b>Command Type</b>	<b>Universe Capability</b>
0000	Interrupt Acknowledge	N/A
0001	Special Cycle	N/A
0010	I/O Read	Slave/Master
0011	I/O Write	Slave Master
0100	Reserved	N/A
0101	Reserved	N/A
0110	Memory Read	Slave/Master
0111	Memory Write	Slave/Master
1000	Reserved	N/A
1001	Reserved	N/A
1010	Configuration Read	Slave/Master
1011	Configuration Write	Slave/Master
1100	Memory Read Multiple	(See Text)
1101	Dual Address Cycle	N/A
1110	Memory Read Line	(See Text)
1111	Memory Write and Invalidate	(See Text)

Memory Read Multiple and Memory Read Line transactions are aliased to Memory Read transactions when the Universe is accessed as a PCI target with these commands. Likewise, Memory Write and Invalidate is aliased to Memory Write. As a PCI initiator, the Universe never generates these commands.

PCI slaves are expected to assert DEVSEL# if they have decoded the access. During a Configuration cycle, the slave is selected by its particular IDSEL#. If a target does not respond with DEVSEL# within 6 clocks, a master abort is generated. The role of configuration cycles is described in the PCI 2.1 Specification.

### 2.3.1.4 Data Transfer

The acknowledgment for a data phase for a transaction occurs on the first rising clock edge after both IRDY# and TRDY# are asserted by the master and slave, respectively. REQ64# may be driven during the address phase to indicate that the master wishes to initiate a 64-bit transaction. The PCI slave asserts ACK64# if it is able to respond to the 64-bit transaction.

Wait cycles are introduced by either the master or the slave by deasserting IRDY# or TRDY#. For write cycles, data is valid on the first rising edge after IRDY# is asserted. Data is acknowledged by the slave on the first rising edge with TRDY# asserted. For read cycles, data is transferred and acknowledged on first rising edge with both IRDY# and TRDY# asserted.

A single data transfer cycle is repeated every time IRDY# and TRDY# are both asserted. The transaction only enters the termination phase when FRAME# is deasserted (master initiated termination) or if STOP# is asserted (slave initiated). When both FRAME# and IRDY# are deasserted (final data phase is complete), the bus is defined as idle.

### 2.3.1.5 Termination Phase

The Universe PCI bus interface permits all four types of PCI terminations:

1. Master-Abort: the PCI bus master negates FRAME# when no slave responds (DEVSEL# not asserted) after 6 clock cycles.
2. Target-Disconnect: a termination is requested by the slave (STOP# is asserted) because it is unable to respond within the latency requirements of the PCI specification or it requires a new address phase. Target-disconnect means that the transaction is terminated after data is transferred. The Universe will deassert REQ# for at least two clock cycles if it receives STOP# from the PCI slave.
3. Target-Retry: termination is requested (STOP# is asserted) by the slave because it cannot currently process the transaction. Retry means that the transaction is terminated after the address phase without any data transfer.
4. Target-Abort: is a modified version of target-disconnect where the slave requests a termination (asserts STOP#) of a transaction which it will never be able to respond to, or during which a fatal error occurred. Although there may be a fatal error for the initiating application, the transaction completes gracefully, ensuring normal PCI operation for other PCI resources.

### 2.3.1.6 Parity Checking

The Universe both monitors and generates parity information on the PAR# signal. The Universe monitors PAR# when it accepts data as a master during a read or a slave during a write. The Universe drives PAR# when it provides data as a slave during a read or a master during a write. The Universe also drives PAR# during the address phase of a transaction when it is a master and monitors PAR# during an address phase when it is the PCI slave. In both address and data phases, the PAR# signal provides even parity for C/BE#[7:0] and AD[63:0]. The Universe continues with a transaction independent of any parity errors reported during the transaction.

The Universe can also be programmed to report address parity errors. It does this by asserting the SERR# signal and setting a status bit in its registers. No interrupt is generated, and regardless of whether assertion of SERR# is enabled, the Universe does not respond to the errored access. If powered up in a 64-bit PCI environment, the Universe uses PAR64 in the same way as PAR, except for AD[63:32] and C/BE[7:4].

### 2.3.2 Universe as PCI Master

The Universe requests PCI bus mastership through its PCI Master Interface. The PCI Master Interface is available to either the VME Slave Channel (access from a remote VMEbus master) or the DMA Channel.

The VME Slave Channel makes an internal request for the PCI Master Interface when:

- the RXFIFO contains a complete transaction,
- sufficient data exists in the RXFIFO to generate a transaction of length defined by the programmable aligned burst size (PABS), or
- there is a coupled cycle request.

The DMA Channel makes an internal request for the PCI Master Interface when:

- the DMAFIFO has sufficient room to queue an entire transaction (of length defined by PABS) during reads from the PCI bus
- the DMAFIFO has queued an entire transaction (of length defined by PABS) and is ready to write to the PCI bus, or
- the DMA block is completely queued during a write to the PCI bus.

Arbitration between the two channels for the PCI Master Interface follows a round robin protocol. Each channel is given access to the PCI bus for a single transaction. Once that transaction completes, ownership of the PCI Master Interface is granted to the other channel if it requires the bus. The VME Slave Channel and the DMA Channel each have a set of rules that determine when it is ‘done’ with the PCI Master Interface. The VME Slave Channel is done under the following conditions:

- an entire transaction (no greater in length than the programmed aligned burst size) is emptied from the RXFIFO, or
- the coupled cycle is complete.

The DMA Channel is done when:

- the boundary programmed into the PCI aligned burst size is emptied from the DMAFIFO during writes to the PCI bus, or
- the boundary programmed into the PCI aligned burst size is queued to the DMAFIFO during reads from the PCI bus.

As discussed elsewhere (“Universe as VME Slave” on page 2-12), access from the VMEbus may be either coupled or decoupled. For a full description of the operation of these data paths, see “Universe as VME Slave” on page 2-12.

The Universe PCI Master Interface can generate the following command types:

- I/O Read,
- I/O Write,
- Memory Read,
- Memory Write,
- Configuration Read (Type 0 and 1), and
- Configuration Write (Type 0 and 1).

The type of cycle the Universe generates on the PCI bus depends upon which VMEbus slave image is accessed and how it is programmed. For example, one slave image might be programmed as an I/O space and another as memory space (see “VME Slave Images” on page 2-44). When generating a memory transaction, the implied addressing is either 32-bit or 64-bit aligned, depending upon the PCI slave. When generating an I/O transaction, the implied addressing is 32-bit aligned and all incoming transactions are coupled.

Both Type 0 and Type 1 cycles are generated and handled through the same mechanism. Once a VME cycle is received and mapped to a configuration cycle, the Universe compares bits [23:16] of the incoming address with the value stored in the Universe’s MAST\_CTL Register’s Bus Number field (BUS\_NO[7:0] in Table A.56). If the bits are the same as the BUS\_NO field, then a TYPE 0 access is generated. If they are not the same, a Type 1 configuration access is generated. The PCI bus-generated address then becomes an unsigned addition of the incoming VMEbus address and the VME slave image translation offset.

### 2.3.2.1 PCI Burst Transfers

The Universe generates aligned burst transfers of some maximum alignment, according to the programmed PCI aligned burst size. The PCI aligned burst size can be programmed at 32 or 64 bytes. Burst transfers will not cross 32-byte or 64-byte boundaries, depending on the programmed alignment (PABS bit in the MAST\_CTL register, Table A.56). For example, when programmed for 32-byte boundaries, a new burst will begin at XXXX\_XX20, XXXX\_XX40, etc. If necessary, a new burst will begin at an address with the programmed alignment. To optimize PCI bus usage, the Universe always attempts to transfer data in aligned bursts at the full negotiated width of the PCI bus.

The Universe can perform a 64-bit data transfer over the AD [63:0] lines, if operated in a 64-bit PCI environment or against a 64-bit capable target or initiator. The LD64EN bit must be set if the access is being made through a VMEbus slave image; the LD64EN bit must be set if the access is being performed with the DMA.

The Universe generates burst cycles on the PCI bus if it is:

- emptying the RXFIFO (the RXFE status bit in the MISC\_STAT register is set when the RXFIFO empties),
- filling the RDFIFO (receives a block read request from a VMEbus master to an appropriately programmed VMEbus slave image), or
- performing DMA transfers

All other accesses are treated as single data beat transactions on the PCI bus.

During PCI burst transactions, the Universe dynamically enables byte lanes on the PCI bus by changing the BE# signals during each data phase.

### 2.3.2.2 Termination

The Universe performs a master abort if the slave does not respond within 6 clock cycles. Coupled PCI transactions terminated with target abort or master abort are terminated on the VMEbus with BERR\*. The R\_TA or R\_MA bits in the PCI\_CS register (Table A.3) are set when the Universe receives a target abort or generates a master abort independent of whether the transaction was coupled, decoupled, prefetched, or initiated by the DMA.

If the Universe receives a retry from the PCI slave, then it relinquishes the PCI bus and re-requests within 2-3 PCI clock cycles. No other transactions are processed by the PCI Master Interface until the retry condition is cleared. The Universe can be programmed to perform a maximum number of retries using the MAXRTRY field in the MAST\_CTL register (Table A.56). When this number of retries has been reached, the Universe responds in the same way as it does to a target abort on the PCI bus. That is, the Universe may issue a BERR\* signal on the VMEbus. Target aborts are discussed in the next two paragraphs. All VME slave coupled transactions and decoupled transactions will encounter a delayed DTACK once the FIFO fills until the condition clears either due to success or a retry time-out.

If the error occurs during a posted write to the PCI bus (see also “Bus Error Handling” on page 2-52), the Universe uses the L\_CMDERR register (Table A.31) to log the command information for the transaction (CMDERR [3:0]) and the address of the errored transaction is latched in the LAERR register (Table A.32). The L\_CMDERR register also records if multiple errors occur (with the M\_ERR bit) although the number of errors is not given. The error log is qualified with the L\_STAT bit. The rest of the transaction will be purged from the RXFIFO if some portion of the write encounters an error. An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and LERR interrupts are enabled (see “The Interrupt Channel” on page 2-56).

If an error occurs during a prefetched read, it is only translated to the VMEbus (as a BERR\* signal) during the first data beat (the coupled portion) of the prefetched read. A target abort on subsequent data beats during a prefetched read is ignored, but does cause prefetching to stop. There is no logging of the error, and all previously fetched data is provided to the VMEbus master. Once the block read on the VMEbus extends beyond the available data in the RDFIFO, then a new prefetch is initiated on the PCI bus. The point at which the new prefetch is initiated corresponds to where the error previously occurred. If the error occurs again, it will be on the first (coupled) data beat of the transaction and will be translated to the VMEbus as a BERR\* signal. As with all coupled errors, no error is logged and no interrupt is generated.

### 2.3.2.3 Parity

The Universe monitors PAR# when it accepts data as a master during a read and drives PAR# when it provides data as a master during a write. The Universe also drives PAR# during the address phase of a transaction when it is a master. In both address and data phases, the PAR# signal provides even parity for C/BE#[3:0] and AD[31:0]. If the Universe is powered up in a 64-bit PCI environment, then PAR64 provides even parity for C/BE#[7:4] and AD[63:32].

The PERESP bit in the PCI\_CS register (Table A.3) determines whether or not the Universe responds to parity errors as PCI master. Data parity errors are reported through the assertion of PERR# if the PERESP bit is set. Regardless of the setting of these two bits, the D\_PE (Detected Parity Error) bit in the PCI\_CS register is set if the Universe encounters a parity error as a master. The DP\_D (Data Parity Detected) bit in the same register is only set if parity checking is enabled through the PERESP bit and the Universe detects a parity error while it is PCI master (i.e. it asserts PERR# during a read transaction or receives PERR# during a write).

No interrupts are generated by the Universe in response to parity errors reported during a transaction. Parity errors are reported by the Universe through assertion of PERR# and by setting the appropriate bits in the PCI\_CS register. If PERR# is asserted to the Universe while it is PCI master, the only action it takes is to set the DP\_D. The Universe continues with a transaction independent of any parity errors reported during the transaction.

As a master, the Universe does not monitor SERR#. It is expected that a central resource on the PCI bus will monitor SERR# and take appropriate action.

### 2.3.3 Universe as PCI Slave

The Universe becomes PCI bus slave when one of its four programmed PCI slave images or one of its registers is accessed by a PCI bus master (the Universe cannot be that PCI bus master). Register accesses are discussed elsewhere (see “Registers” on page 2-87); this section describes only those accesses destined for the VMEbus.

When one of its PCI slave images is accessed, the Universe responds with DEVSEL# within two clocks of FRAME# (making the Universe a medium speed device, as reflected by the DEVSEL field in the PCI\_CS register).

As PCI slave, the Universe responds to the following command types:

- I/O Read,
- I/O Write,
- Memory Read,
- Memory Write,
- Configuration Read (Type 1),
- Configuration Write (Type 1),
- Memory Read Multiple (aliased to Memory Read),
- Memory Line Read (aliased to Memory Read),
- Memory Write and Invalidate (aliased to Memory Write).

Type 0 configuration accesses can only be made to the Universe's PCI configuration registers. The PCI slave images do not accept Type 0 accesses; they only accept Type 1 accesses if AD[23:16] matches the BUS\_NO field in the MAST\_CTL register.

Address parity errors are reported if both PERESP and SERR\_EN are set in the PCI\_CS register (Table A.3). Address parity errors are reported by the Universe by asserting the SERR# signal and setting the S\_SERR (Signalled SERR#) bit in the PCI\_CS register. Assertion of SERR# can be disabled by clearing the SERR\_EN bit in the PCI\_CS register. No interrupt is generated, and regardless of whether assertion of SERR# is enabled or not, the Universe does not respond to the access with DEVSEL#. Typically the master of the transaction times out with a master-abort.

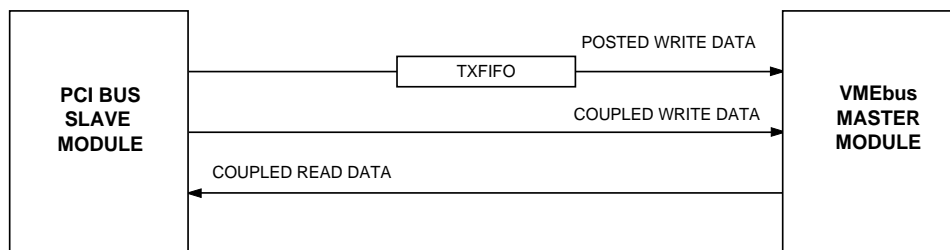
If the Universe is accessed with REQ64# as a 64-bit target, then it responds with ACK64# if it is powered up as a 64-bit device.

If the VME Cycle Type field (VCT) of the PCI Slave Image Control Register (LSIx\_CTL in Table A.8 or Table A.12) corresponding to the incoming AD signal is set to binary 10, then the incoming AD[23:16] is compared to the BUS\_NO field in the MAST\_CTL Register (Table A.56). If they match then the cycle is accepted and a VMEbus cycle is generated.

### 2.3.3.1 Data Transfer

Read transactions are always coupled (as opposed to VMEbus slave reads, which may be pre-fetched; see "Universe as VME Slave" on page 2-12). Write transactions can be coupled or posted (see Figure 2.5 below and "PCI Bus Slave Images" on page 2-47). To ensure sequential consistency, coupled operations (reads or writes) are only processed once all previously posted write operations have completed (i.e. the TXFIFO is empty).





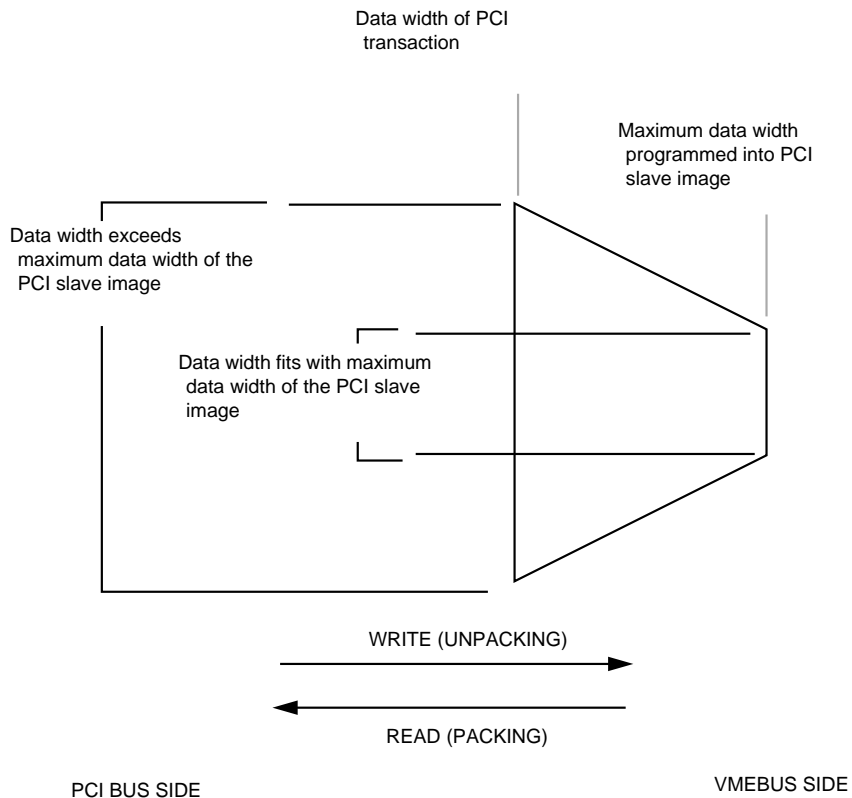
**Figure 2.5 : PCI bus Slave Channel Dataflow**

The data transfer between the PCI bus and VMEbus is perhaps best explained by Figure 2.6 below. The Universe can be seen as a funnel where the mouth of the funnel is the data width of the PCI transaction. The end of the funnel is the maximum VMEbus data width programmed into the PCI slave image (VDW bit in the PCI slave image control register). For example, consider a 32bit PCI transaction accessing a PCI slave image with VDW set to 16 bits. A data beat with all byte lanes enabled will be broken into two 16-bit cycles on the VMEbus. If the PCI slave image is also programmed with block transfers enabled, the 32-bit PCI data beat will result in a D16 block transfer on the VMEbus. Write data is unpacked to the VMEbus and read data is packed to the PCI bus data width.

If the data width of the PCI data beat is the same as the maximum data width of the PCI slave image, then the Universe maps the data beat to an equivalent VMEbus cycle. For example, consider a 32bit PCI transaction accessing a PCI slave image with VDW set to 32 bits. A data beat with all byte lanes enabled is translated to a single 32-bit cycle on the VMEbus.

As the general rule, if the PCI bus data width is less than the VMEbus data width then there is no packing or unpacking between the two buses. The only exception to this is during 32-bit PCI multi-data beat transactions to a PCI slave image programmed with maximum VMEbus data width of 64 bits. In this case, packing/unpacking occurs to make maximum use of the full bandwidth on both buses.

Only aligned VMEbus transactions are generated, so if the requested PCI data beat has unaligned or non-contiguous byte enables, then it is broken into multiple aligned VMEbus transactions no wider than the programmed VMEbus data width. For example, consider a three-byte PCI data beat (on a 32-bit PCI bus) accessing a PCI slave image with VDW set to 16 bits. The three-byte PCI data beat will be broken into two aligned VMEbus cycles: a single-byte cycle and a double-byte cycle (the ordering of the two cycles depends on the arrangement of the byte enables in the PCI data beat). If in the above example the PCI slave image has a VDW set to 8 bits, then the three-byte PCI data beat will be broken into three single-byte VMEbus cycles.



**Figure 2.6 : Influence of Transaction Data Width and Slave Image Data Width on Data Packing/Unpacking**

### 2.3.3.2 Coupled Transfers

By default, all PCI slave images are set for coupled transfers. If a data phase of a coupled burst requires packing or unpacking on the VMEbus, acknowledgment of the transfer is not received on the PCI bus until all data has been packed or unpacked on the VMEbus. If an error occurs during any of the packing or unpacking, then the transaction is terminated on the PCI bus with Target-Abort. Target retries will be generated by the Universe while there are posted writes in the TXFIFO or while the PCI Slave Channel does not have VMEbus ownership. Note that the Universe may be the owner of the VMEbus through the DMA or Interrupt Channels in which case the PCI master will be retried. See “VMEbus Release” on page 2-8 for more details on how the Universe shares the VMEbus between channels.

PCI latency requirements (as described in revision 2.0 of the PCI Specification) require that only 8 clock cycles can elapse between the first and second data beat of a transaction. Since the Universe cannot guarantee that data acknowledgment will be received from the VMEbus in time to meet these PCI latency requirements, the Universe performs a target-disconnect after the first data beat of every coupled transaction.

#### Retried Coupled Transfers

As PCI slave, the Universe retries all coupled VMEbus accesses unless the Local Slave Channel is the current owner of the VMEbus. This prevents deadlock conditions and improves the utilization of the PCI bus (i.e. increases the probability that a coupled transfer will proceed as soon as possible without blocking all other PCI bus traffic).

After initiating a target-retry to a coupled access, the Universe requests the VMEbus. While VMEbus ownership is pending, it continues to retry all PCI masters, both coupled and decoupled, except for those accessing the Universe registers. Once VMEbus ownership is obtained, any coupled request from a PCI master is mapped directly to the VMEbus.

The Universe has a configurable Coupled Request Timer (set with the CRT field in the LMISC register, Table A.29) that is restarted every time a coupled request is retried. The CRT can be programmed to 128  $\mu$ s, 256  $\mu$ s, 512  $\mu$ s, 1024  $\mu$ s, 2048  $\mu$ s, 4096  $\mu$ s, or disabled. By default, the CRT is set to 128  $\mu$ s. If set to 0, the VMEbus is held until a coupled request arrives. If the Universe obtains the VMEbus and receives no request for coupled access before the Coupled Request Timer expires, then it will release the VMEbus.

#### Consecutive Coupled Transfers

The Universe has a programmable coupled window timer that allows consecutive coupled transactions to occur without the need to release the VMEbus between transactions. After completing a coupled transaction, the Universe delays its release of the VMEbus by the amount of time programmed into the coupled window timer. The timer can be set with the CWT field in the LMISC register (Table A.29) to count PCI clocks. It can be programmed to

16, 32, 64, 128, 256, or 512 clocks. By default the CWT is disabled so that VMEbus ownership is relinquished immediately by the PCI Slave Channel after each coupled cycle. If the release mode is set for RWD or there is an external VME request pending in ROR mode, then this causes VMEbus ownership to be relinquished by the Universe independent of other channel requests.

If another coupled transaction is generated during the timer period, it will be processed directly and will not be retried. If another cycle is not attempted before the window expires, then the PCI Slave Channel releases its ownership of the VMEbus.

### 2.3.3.3 Posted Writes

Posted writes are enabled for a PCI slave image by setting the PWEN bit in the control register of the PCI Slave Image (see “PCI Bus Slave Images” on page 2-47). Write transactions are relayed from the PCI bus to the VMEbus through a 18-entry deep TXFIFO. The TXFIFO allows each entry to contain 32 address bits (with extra bits provided for command information), or up to 64 data bits. For each posted write transaction received from the PCI bus, the PCI Slave Interface queues an address entry in the FIFO. This entry contains the translated address space and mapped VMEbus attributes information relevant to the particular PCI slave image that has been accessed (see “PCI Bus Slave Images” on page 2-47). For this reason, any re-programming of PCI bus slave image attributes will only be reflected in TXFIFO entries queued after the re-programming. Transactions queued before the re-programming are delivered to the VMEbus with the PCI bus slave image attributes that were in use before the re-programming.



*Caution: Care should be taken before reprogramming slave images from one bus while that image is being accessed from the opposite bus. If there is a chance the image may be accessed while being reprogrammed, disable the image first before changing image attributes.*

Once the address phase is queued in one TXFIFO entry, the PCI Slave Interface may pack the subsequent data beats to a full 64-byte width before queuing the data into new entries in the TXFIFO.

The number of data beats that the PCI Slave Channel queues in the TXFIFO for a single posted write depends upon the programming of the PCI aligned burst size field (PABS) in the MAST\_CTL register (Table A.56). The PABS field is programmable to 32 or 64 bytes (coupled transactions are not affected by the PABS field and are always broken up after the first data beat, see above). When a PCI master attempts a posted write transaction that crosses the programmed aligned burst boundary, the Universe’s PCI Slave Interface generates a target disconnect to the master at the aligned burst boundary. The transaction up to the aligned burst boundary is queued in the TXFIFO as one address entry plus up to four or eight data entries (32 or 64 bytes) depending on the programming of PABS.

To increase performance and bus utilization on the PCI bus, the Universe does not queue a transaction into the TXFIFO until the TXFIFO contains enough space for a full aligned burst (as defined by PABS). If the TXFIFO does not have enough space for an aligned burst, then the PCI posted write transaction is terminated by the Universe with a target-retry immediately after the address phase. This mechanism, where the PCI Slave Interface only accepts transactions when it has sufficient room for a full aligned burst, ensures that the Universe always accepts a full transaction (as defined by the programmed aligned burst size) without inserting wait states or terminating a transaction early.

Additionally, although there are 18 entries available in the Tx FIFO, a maximum of four complete transactions may be queued at any one time. If four transaction have already been queued, the PCI Slave Interface retries all subsequent accesses until a transaction has been delivered to the VMEbus.

Before a transaction can be delivered to the VMEbus from the TXFIFO, the PCI Slave Channel must obtain ownership of the VME Master Interface. Ownership of the VME Master Interface is granted to the different channels on a round robin basis (see “VMEbus Release” on page 2-8). Once the PCI Slave Channel obtains the VMEbus through the VME Master Interface, the manner in which the TXFIFO entries are delivered depends upon the programming of the VMEbus attributes in the PCI slave image (see “PCI Bus Slave Images” on page 2-47). For example, if the VMEbus data width is programmed to 16 bits, and block transfers are disabled, then each data entry in the TXFIFO corresponds to four transactions on the VMEbus.

If block transfers are enabled in the slave image, BLTs or MBLTs are generated on the VMEbus such that each transaction queued in the TXFIFO corresponds to a single VMEbus block transfer. This means that unlike the DMA Channel (which can generate full block sizes of 256 bytes for BLTs and 2 Kbytes for MBLTs), the PCI Slave Channel can generate block transfers only at the programmed aligned burst size (32 or 64 bytes, PABS in the MAST\_CTL register, Table A.56). This corresponds (depending upon the programming of PABS) to eight or sixteen data phases per BLT; four or eight data phases per MBLT. In addition, note that if block transfers are enabled in the PCI slave image, then each transaction queued in the TXFIFO, independent of its length, is delivered to the VMEbus as a block transfer. This means that if a single data beat transaction is queued in the TXFIFO, it appears on the VMEbus as a single data phase block transfer.

Any PCI master attempting coupled transactions is retried while the TXFIFO contains data.

### 2.3.3.4 RMW and ADOH Cycles

The Universe will only generate RMW and ADOH cycles on the VMEbus through the use of a Special Cycle Generator. The Special Cycle Generator is configured through the register fields shown in Table 2.3 below.

**Table 2.3 : Register Fields for the Special Cycle Generator**

Field	Register Bits	Description
32-bit address	ADDR in Table A.25	specifies PCI bus slave image address
type	SCYC in Table A.24	disabled, RMW or ADOH
32-bit enable	EN [31:0] in Table A.26	a bit mask to select the bits to be modified in the VMEbus read data during a RMW cycle
32-bit compare	CMP [31:0] in Table A.27	data which is compared to the VMEbus read data during a RMW cycle
32-bit swap	SWP [31:0] in Table A.28	data which is swapped with the VMEbus read data and written to the original address during a RMW cycle

The 32-bit address field is used to select an address within one of the currently enabled A32 PCI slave images. When the SCYC field is programmed, the VMEbus transaction parameters associated with the selected VMEbus address are temporarily disabled, although the translation offset for the slave image still applies. Instead, the VMEbus transaction parameters used in the transaction are those specified in the Special Cycle Generator registers. When a PCI master accesses the selected VMEbus address while the SCYC field is set, a coupled cycle of the programmed type (RMW or ADOH) is executed on the VMEbus. Either a read or write to the selected address will initiate the RMW or ADOH transaction (the different effects of a read or write are described in the separate RMW and ADOH sections below). When the VMEbus transaction is complete (and the SCYC field is cleared), the selected transaction parameters belonging to the PCI address return to those defined in the PCI slave image.

The following sections describe the specific properties for each of the transfer types: RMW and ADOH.

#### Read-Modify-Write

When the SCYC field is set to RMW, any read access to the specified VMEbus address will result in a RMW cycle on the VMEbus. The data from the read portion of the RMW on the VMEbus is returned as the read data on the PCI bus. If initiated with a write transaction, a standard write occurs with the attributes programmed in the PCI slave image.

RMW cycles make use of three 32-bit registers. The bit enable field is a bit mask which lets the user specify which bits in the read data are compared and modified in the RMW cycle. This bit enable setting is completely independent of the RMW cycle data width, which is determined by the data width of the initiating PCI transaction. During a RMW, the VMEbus read data is bitwise compared with the SCYC\_CMP and SCYC\_EN registers. The valid compared and enabled bits are then swapped using the SCYC\_SWP register.

Each enabled bit that compares true is swapped with the corresponding bit in the 32-bit swap field. A false comparison results in the original bit being written back.

Once the RMW cycle completes, the VMEbus read data is returned to the waiting PCI bus master and the PCI cycle terminates.

### ADOH

When the SCYC field is set to ADOH, any write access to the specified VMEbus address will result in an ADOH cycle on the VMEbus. Reads to the specified address translates to VMEbus reads in the standard fashion. The data during writes is ignored. The AM code generated on the VMEbus is determined by the PCI slave image definition for the specified VMEbus address (see Table 2.8 on page 47).

The ADOH cycle is used to implement the VMEbus Lock command. However, after the ADOH cycle is complete, there is no guarantee that the Universe will remain VMEbus master. If the Universe loses VMEbus ownership, then the VMEbus Lock command will no longer apply. The user can use the VMEbus ownership described in “Exclusive Accesses” on page 2-41 to ensure that the Universe will maintain VMEbus ownership after the ADOH cycle is complete.

#### 2.3.3.5 Exclusive Accesses

A PCI master obtains exclusive access to a resource on the VMEbus through the combined use of the PCI LOCK# signal (which locks the PCI Slave Interface against other PCI masters) and the VMEbus ADOH cycle (which locks the VMEbus resource against other VMEbus masters). LOCK# ensures the PCI master sole PCI access to the Universe registers and the VMEbus. The ADOH cycle ensures that the VME Master Interface has exclusive access to the VMEbus resource while it has VMEbus tenure. If there is only one master on the PCI bus, there is no need for the PCI lock.

To further guarantee exclusive access, the Universe provides a VMEbus ownership bit (VOWN bit in the MAST\_CTL register, Table A.56) to ensure that the Universe has access to the locked VMEbus resource for an indeterminant period. The Universe can be programmed to assert an interrupt on the PCI bus when it acquires the VMEbus and the VOWN bit is set (VOWN enable bit in the LINT\_EN register, Table A.40). While the VMEbus is held using the VOWN bit, the Universe sets the VOWN\_ACK bit in the MAST\_CTL register. The VME Master Interface maintains bus tenure while the ownership bit is set, and only releases the VMEbus when the ownership bit is cleared. This function is important for the following two reasons.

If the VME Master Interface is programmed for RWD, it may release the VMEbus when the PCI Slave Channel has completed a transaction (definition of ‘done’ for the PCI Slave Channel, see “VMEbus Release” on page 2-8). Therefore, if exclusive access to the VMEbus resource is required for multiple transactions, then the VMEbus ownership bit will hold the bus until the exclusive access is no longer required.

Alternatively, if the VME Master Interface is programmed for ROR, the VMEbus ownership bit will ensure VMEbus tenure even if other VMEbus requesters require the VMEbus.



*Posted writes will be treated as coupled transactions while the VMEbus ownership bit is set.*



### 2.3.3.6 Terminations

The Universe performs the following terminations as PCI slave:

1. Target-Disconnect
  - when registers are accessed (no bursts allowed to registers),
  - after the first data beat of every coupled cycle,
  - when the transaction reaches a programmable aligned burst size boundary.
2. Target-Retry
  - when a PCI master tries to write to the TXFIFO when there is insufficient room for a full burst,
  - when a coupled transaction is attempted and the Universe does not own the VMEbus, or
  - when a coupled transaction is attempted while the TXFIFO has entries to process.
3. Target-Abort
  - when the Universe receives BERR\* on the VMEbus during a coupled cycle (BERR\* translated as Target-Abort on the PCI side and the S\_TA bit is set in the PCI\_CS register, Table A.3).

Whether to terminate a transaction or for retry purposes, the Universe keeps STOP# asserted until FRAME# is deasserted, independent of the logic levels of IRDY# and TRDY#. If STOP# is asserted while TRDY# is deasserted, it means that the Universe will not transfer any more data to the master.

If an error occurs during a posted write to the VMEbus, the Universe uses the V\_AMERR register (Table A.80) to log the AM code of the transaction (AMERR [5:0]), and the state of the IACK\* signal (IACK# bit, to indicate whether the error occurred during an IACK cycle). The FIFO entries for the offending cycle are purged. The V\_AMERR register also records whether multiple errors have occurred (with the M\_ERR bit) although the number is not given. The error log is qualified with the V\_STAT bit (logs are valid if the V\_STAT bit is set). The address of the errored transaction is latched in the VAERR register (Table A.81). When the Universe receives a VMEbus error during a posted write, it generates an interrupt on the VMEbus and/or PCI bus depending upon whether the VERR and VERR interrupts are enabled (see “The Interrupt Channel” on page 2-56).

## 2.4 Slave Image Programming

The Universe recognizes two types of accesses on its bus interfaces: accesses destined for the other bus, and accesses decoded for its own register space. Address decoding for the Universe's register space is described in "Registers" on page 2-87. This section describes the slave images used to map transactions between the PCI bus and VMEbus.

### 2.4.1 VME Slave Images

The Universe accepts accesses from the VMEbus within specific programmed slave images. Each VMEbus slave image opens a window to the resources of the PCI bus and, through its specific attributes, allows the user to control the type of access to those resources. The tables below describe programming for the VMEbus slave images by dividing them into VMEbus, PCI bus and Control fields.

**Table 2.4 : VMEbus Fields for VME Slave Image**

Field	Register Bits	Description
base	BS[31:12] in Table A.61	multiples of 4 or 64 Kbytes (base to bound: maximum of 4 GBytes)
bound	BD[31:12] in Table A.62	
address space	VAS in Table A.60	A16, A24, A32, User 1, User 2
mode	SUPER in Table A.60	supervisor and/or non-privileged
type	PGM in Table A.60	program and/or data

**Table 2.5 : PCI Bus Fields for VME Slave Image**

Field	Register Bits	Description
translation offset	TO[31:12] in Table A.63	offsets VMEbus slave address to a selected PCI address
address space	LAS in Table A.60	Memory, I/O, Configuration
RMW	LLRMW in Table A.60	RMW enable bit

**Table 2.6 : Control Fields for VME Slave Image**

Field	Register Bits	Description
image enable	EN in Table A.60	enable bit
posted write	PWEN in Table A.60	posted write enable bit
prefetched read	PREN in Table A.60	prefetched read enable bit
enable PCI D64	LD64EN in Table A.60	enables 64-bit PCI bus transactions



*Note: If the programming for an image is changed after the transaction is queued in the FIFO, the transaction's attributes are not changed. Only subsequent transactions are affected by the change in attributes.*

#### 2.4.1.1 VMEbus Fields

Decoding for VMEbus accesses is based on the address, and address modifiers produced by the VMEbus master. Before responding to an external VMEbus master, the address must lie in the window defined by the base and bound addresses, and the Address Modifier must match one of those specified by the address space, mode, and type fields.

The Universe's four VME slave images (images 0 to 3) are bounded by A32 space. The first of these (VME slave image 0) has a 4 Kbyte resolution while VME slave images 1 to 3 have 64-Kbyte resolution (maximum image size of 4 GBytes). Typically, image 0 would be used as an A16 image since it provides the finest granularity of the four images.

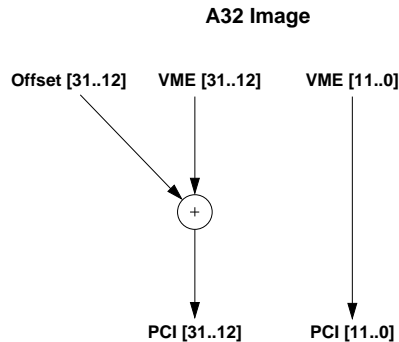


*Note that the address space of a VMEbus slave image must not overlap with the address space for the Universe's control and status registers. Since register accesses take priority, any access to the overlap region results in a register access.*

#### 2.4.1.2 PCI Bus Fields

The PCI bus fields specify how the VMEbus transaction is mapped to the appropriate PCI bus transaction. The translation offset field allows the user to translate the VMEbus address to a different address on the PCI bus. The translation of VMEbus transactions beyond 4 Gbytes results in wrap-around to the low portion of the address range.

The PAS field controls generation of the PCI transaction command. The LLRMW bit allows indivisible mapping of incoming VMEbus RMW cycles to the PCI bus via the PCI LOCK# mechanism (see "VMEbus Read Modify Writes" on page 2-18). When the LLRMW bit is set, single cycle reads will always be mapped to single data beat locked PCI transactions. Setting this bit has no effect on non-block writes: they can be coupled or decoupled. However, note that only accesses to PCI Memory Space are decoupled, accesses to I/O or Configuration Space are always coupled.



**Figure 2.7 : Address Translation Mechanism for VMEbus to PCI Bus Transfers**

### 2.4.1.3 Control Fields

The control fields allow the user to enable a VMEbus slave image (using the EN bit), as well as specify how reads and writes will be processed. At power-up, all images are disabled and are configured for coupled reads and writes.

If the PREN bit is set, the Universe will prefetch for incoming VMEbus block read cycles. It is the user's responsibility to ensure that prefetched reads are not destructive and that the entire image contains prefetchable resources.

If the PWEN bit is set, incoming write data from the VMEbus is loaded into the RXFIFO (see “Posted Writes” on page 2-14). Note that posted write transactions can only be mapped to Memory space on the PCI bus. Setting the PAS bit in the PCI fields to I/O or Configuration Space will force all incoming cycles to be coupled independent of this bit.

If the LD64EN bit is set, the Universe will attempt to generate 64-bit transactions on the PCI bus by asserting REQ64#. The REQ64# line is asserted during the address phase in a 64-bit PCI system, and is the means of determining whether the PCI slave is a 64-bit port. If the slave asserts ACK64# with DEVSEL#, then the Universe uses the 64-bit data bus. If the slave does not assert ACK64# with DEVSEL#, then the Universe uses a 32-bit data bus. However, note that use of REQ64# requires extra clocks internally. Therefore, if no 64-bit targets are expected on the PCI bus then performance can be improved by disabling LD64EN on the VME slave images.



*In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).*

## 2.4.2 PCI Bus Slave Images

The Universe accepts accesses from the PCI bus with specific programmed PCI slave images. Each image opens a window to the resources of the VMEbus and allows the user to control the type of access to those resources. The tables below describe programming for the four standard PCI bus slave images (numbered 0 to 3) by dividing them into VMEbus, PCI bus and Control fields. One special PCI slave image separate from the four discussed below is described in “Special PCI Slave Image” on page 2-49.

**Table 2.7 : PCI Bus Fields for the PCI Bus Slave Image**

Field	Register Bits	Description
base	BS[31:12] in Table A.9	multiples of 4 or 64 Kbytes (base to bound: maximum of 4 GBytes)
bound	BD[31:12] in Table A.10	
address space	LAS in Table A.8	Memory, I/O, Configuration

**Table 2.8 : VMEbus Fields for the PCI Bus Slave Image**

Field	Register Bits	Description
translation offset	TO[31:12] in Table A.11	translates address supplied by PCI master to a specified VMEbus address
maximum data width	VDW in Table A.8	8, 16, 32, or 64 bits
address space	VAS in Table A.8	A16, A24, A32, CR/CSR, User1, User2
mode	SUPER in Table A.8	supervisor or non-privileged
type	PGM in Table A.8	program or data
cycle	VCT in Table A.8	single or block

**Table 2.9 : Control Fields for PCI Bus Slave Image**

Field	Register Bits	Description
image enable	EN in Table A.8	enable bit
posted write	PWEN in Table A.8	enable bit



*Note: If the programming for an image is changed after the transaction is queued in the FIFO, the transaction's attributes are not changed.*

### 2.4.2.1 PCI Bus Fields

All decoding for VMEbus accesses are based on the address and command information produced by a PCI bus master. The Universe slave claims a cycle if there is an address match and if the command matches certain criteria.

All of the Universe's four PCI slave images are A32-capable only. The first of these (PCI slave image 0) has a 4 Kbyte resolution while PCI slave images 1 to 3 have 64 Kbyte resolution. Typically, image 0 would be used for an A16 image since it has the finest granularity.

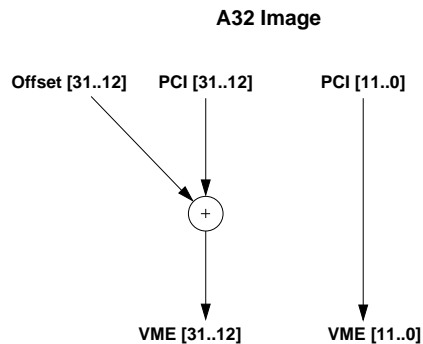


*Note that the address space of a PCI bus slave image must not overlap with the address space for the Universe's control and status registers. Since register accesses take priority, any access to the overlap region results in a register access.*

### 2.4.2.2 VMEbus Fields

The VMEbus fields map PCI transactions to a VMEbus transaction, causing the Universe to generate the appropriate VMEbus address, AM code, and cycle type. Some invalid combinations exist within the PCI slave image definition fields. For example, A16 and CR/CSR spaces do not support block transfers, and A16 space does not support 64-bit transactions. Note that the Universe does not attempt to detect or prevent these invalid programmed combinations, and that use of these combinations may cause illegal activity on the VMEbus.

The 21-bit translation offset allows the user to translate the PCI address to a different address on the VMEbus. The figure below illustrates the translation process:



**Figure 2.8 : Address Translation Mechanism for PCI Bus to VMEbus Transfers**

Translations beyond the 4 Gbyte limit will wrap around to the low address range.

The AM code generated by the Universe is a function of: the VMEbus fields, and the data width and alignment generated by the PCI bus master. For RMW and ADOH cycles, the AM code also depends upon the settings for the Special Cycle Generator (see “RMW and ADOH Cycles” on page 2-40).

The address space, mode, type, and cycle fields control the VMEbus AM code for most transactions. Setting the cycle field to BLT enables the BLT cycle generation. MBLT cycles are generated when the maximum width is set to 64, the BLT bit is set, and the PCI master queues a transaction with at least 64 bits (aligned) of data.

The Universe provides support for user defined AM codes. The USER\_AM register (Table A.59) contains AM codes identified as User1 and User2. If the user selects one of these two, then the corresponding AM code from the global register is generated on the VMEbus. This approach results in standard single cycle transfers to A32 VMEbus address space independent of other settings in the VMEbus fields.

### 2.4.2.3 Control Fields

The control fields allow the user to enable a PCI slave image (the EN bit), as well as specify how writes are processed. If the PWEN bit is set, then the Universe will perform posted writes when that particular PCI slave image is accessed. Posted write transactions are only decoded within PCI Memory space. Accesses from other spaces will result in coupled cycles independent of the setting of the PWEN bit.

### 2.4.3 Special PCI Slave Image

The Universe provides a special PCI slave image located in Memory, I/O, or Configuration space. Its base address is aligned to 64-Mbyte boundaries and its size is fixed at 64 Mbytes (decoded using PCI address lines [31:26]). The Special PCI Slave Image is divided into four 16Mbyte regions numbered 0 to 3 (see Figure 2.9 on page 2-51). These separate regions are selected with PCI address bits AD [25:24]. For example, if AD [25:24] = 01, then region 1 is decoded. Within each region, the upper 64Kbytes map to VMEbus A16 space, while the remaining portion of the 16 Mbytes maps to VMEbus A24 space. Note that no offsets are provided, so address information from the PCI transaction is mapped directly to the VMEbus. The general attributes of each region are programmed according to the tables below.

**Table 2.10 : PCI Bus Fields for the Special PCI Slave Image**

Field	Register Bits	Description
base	BS[5:0] in Table A.30	64 Mbyte aligned base address for the image
address space	LAS [1:0] in Table A.30	Places image in Memory, I/O, or Configuration space

**Table 2.11 : VMEbus Fields for the Special PCI Bus Slave Image**

Field	Register Bits	Description
maximum data width	VDW in Table A.30	separately sets each region for 16 or 32 bits
mode	SUPER in Table A.30	separately sets each region as supervisor or non-privileged
type	PGM in Table A.30	separately sets each region as program or data

**Table 2.12 : Control Fields for the Special PCI Bus Slave Image**

Field	Register Bits	Description
image enable	EN in Table A.30	enable bit for the image
posted write	PWEN in Table A.30	enable bit for posted writes for the image

The special PCI slave image provides access to all of A16 and most of A24 space (all except the upper 64 Kbytes). By using the special PCI slave image for A16 and A24 transactions, it is possible to free the four standard PCI slave images (see “PCI Bus Slave Images” on page 2-47), which are typically programmed to access A32 space.

Note that some address space redundancy is provided in A16 space. The VMEbus specification requires only two A16 spaces, while the special PCI slave image allows for four A16 address spaces.



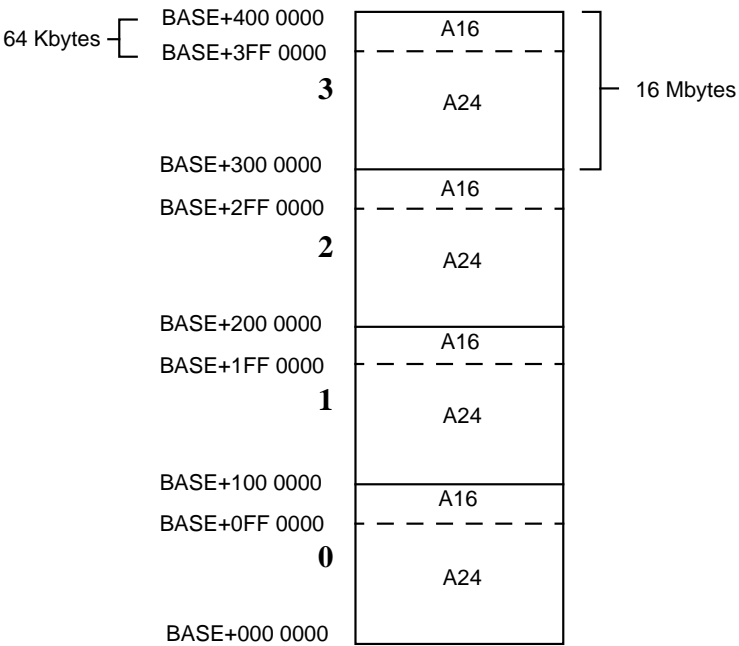


Figure 2.9 : Memory Mapping in the Special PCI Slave Image

## 2.5 Bus Error Handling

There are two fundamentally different conditions under which bus errors may occur with the Universe: during coupled cycles or during decoupled cycles. In a coupled transaction, the completion status is returned to the transaction master, which may then take some action. However, in a decoupled transaction, the master is not involved in the data acknowledgment at the destination bus and higher level protocols are required.

The error handling provided by the Universe is described for both coupled and decoupled transactions below.

### 2.5.1 Coupled Cycles

During coupled cycles, the Universe provides immediate indication of an errored cycle to the originating bus. VMEbus to PCI transactions terminated with target abort or master abort are terminated on the VMEbus with BERR\*. The R\_TA or R\_MA bits in the PCI\_CSR register (Table A.3 on page A-7) are set when it receives a target abort or master abort. For PCI to VMEbus transactions, a VMEbus BERR\* received by the Universe is communicated to the PCI master as a target abort and the S\_TA bit is set (Table A.3). No information is logged in either direction nor is an interrupt generated.

### 2.5.2 Decoupled Transactions

#### 2.5.2.1 Posted Writes

The Universe provides the option of performing posted writes in both the PCI Slave Channel and the VME Slave Channel. Once data is written into the RXFIFO or TXFIFO by the initiating master (VMEbus or PCI bus respectively), the Universe provides positive acknowledgment of the cycle's completion. When the data in the FIFO is written to the destination slave by the Universe, it may subsequently receive a bus error instead of a positive acknowledgment. The Universe handles this situation by logging the errored transactions in one of two error logs and generating an interrupt. Each error log (one for VMEbus errors and one for PCI bus errors) is comprised of two registers: one for address and one for command or address space logging.

If the error occurs during a posted write to the VMEbus, the Universe uses the V\_AMERR register (Table A.80) to log the AM code of the transaction (AMERR [5:0]), and the state of the IACK\* signal (IACK# bit), to indicate whether the error occurred during an IACK cycle. The address of the errored transaction is latched in the V\_AERR register (Table A.81). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the VERR and VERR interrupts are enabled (see “The Interrupt Channel” on page 2-56). The remaining entries of the offending transaction are purged.

If the error occurs during a posted write to the PCI bus, the Universe uses the `L_CMDERR` register (Table A.31) to log the command information for the transaction (`CMDERR [3:0]`). The address of the errored transaction is latched in the `L_AERR` register (Table A.32). An interrupt is generated on the VMEbus and/or PCI bus depending upon whether the `VERR` and `LERR` interrupts are enabled (see “The Interrupt Channel” on page 2-56).

Under either of the above conditions (VME to PCI, or PCI to VME), the address that is stored in the log represents the most recent address the Universe generated before the bus error was encountered. For single cycle transactions, the address represents the address for the actual errored transaction. However, for multi-data beat transactions (block transfers on the VMEbus or burst transactions on the PCI bus) the log only indicates that an error occurred somewhere after the latched address. For a VMEbus block transfer, the logged address will represent the start of the block transfer. Since during PCI Slave Channel activity the Universe generates block transfers no larger than that programmed in the `PABS` field in the `MAST_CTL` register, the error will have occurred from the logged address up to the next aligned address. Similarly for a PCI burst transaction, the error will have occurred anywhere from the logged address up to the next burst aligned address.

In the case of PCI-initiated transactions, all data from the errored address up to the end of the initiating transaction is flushed from the `TXFIFO`. Since the Universe breaks PCI transactions at burst aligned addresses, the data is not flushed past this point. If the PCI master is generating bursts shorter than the Universe’s aligned burst value (as set by the `PABS` in `MAST_CTL` register), then only data up to the end of that transaction is flushed).

Similarly in a posted write from the VMEbus, all data subsequent to the error in the transaction is flushed from the `RXFIFO`. However, the length of a VMEbus transaction differs from the length of the errored PCI bus transaction. For non-block transfers, the length always corresponds to one so only the errored data beat is flushed. However, if an error occurs on the PCI bus during a transaction initiated by a VMEbus block transfer, all data subsequent to the errored data beat in the block transfer is flushed from the `RXFIFO`. In the case of BLTs, this implies that potentially all data up to the next 256-byte boundary may be flushed. For MBLTs, all data up to the next 2-KByte boundary may be flushed.

Once an error is captured in a log, that set of registers is frozen against further errors until the error is acknowledged. The log is acknowledged and made available to latch another error by clearing the corresponding status bit in the `VINT_STAT` or `LINT_STAT` registers. Should a second error occur before the CPU has the opportunity to acknowledge the first error, another bit in the logs is set to indicate this situation (`M_ERR` bit).

### 2.5.2.2 Prefetched Reads

In response to a block read from the VMEbus, the Universe initiates prefetching on the PCI bus (if the VMEbus slave image is programmed with this option). The transaction generated on the PCI bus is an aligned memory read transaction with multiple data beats extending to the aligned burst boundary (as programmed by PABS in the MAST\_CTL register). Once an acknowledgment is given for the first data beat, an acknowledgment is sent to the VMEbus initiator by the assertion of DTACK\*. Therefore, the first data beat of a prefetched read is coupled while all subsequent reads in the transaction are decoupled.

The Universe only translates errors from the PCI bus to the VMEbus during the first data beat (the coupled portion) of the prefetched read. A target abort on the PCI bus in this data beat is translated as a BERR\* signal on the VMEbus. A target abort on subsequent data beats during a prefetched read is ignored, but does cause prefetching to stop. There is no logging of the error, and all previously fetched data is provided to the VMEbus master. Once the block read on the VMEbus extends beyond the available data in the RDFIFO, then a new prefetch is initiated on the PCI bus. The point at which the new prefetch is initiated corresponds to where the error previously occurred. If the error occurs again, it will be on the first (coupled) data beat of the transaction and will be translated to the VMEbus as a BERR\* signal. As with all coupled errors, no error is logged and no interrupt is generated. It is important to recognize that the target may be required to provide two target aborts before it is passed back to the VMEbus. Similarly, a target abort may never be passed back to the VMEbus if the block transfer does not extend to the point at which the error occurred.

### 2.5.2.3 DMA Errors

If bus errors occur during part of a DMA initiated transaction, the DMA halts, generates an interrupt (if the interrupt is enabled), and stops the address and count registers to give an indication of where the error occurred.

What the DMA does next depends upon whether the error occurs on the source bus (during the read) or on the destination bus (during the write). If the bus error occurs on the source bus, all DMA operations on that bus terminate immediately but any data previously queued within the DMAFIFO is still written to the destination bus. Once the DMAFIFO empties, the DMA generates an interrupt (as enabled by INT\_LERR and INT\_VERR in the DMA\_GCS register, Table A.38). The bus on which the interrupt appears is determined by the DMA bits in the LINT\_EN and VINT\_EN registers (Table A.40 and Table A.44). The VERR (for VMEbus error) and LERR (for PCI bus error) bits in the DMA GCS register indicate which bus the error occurred on.

If the error occurs on the destination bus, the DMA immediately halts any operations on that bus, but continues reading from the source bus until the DMAFIFO no longer has room for another transaction. After the DMA stops on the source bus (or all data relevant to the command packet is read), the error bit in the DMA\_GCSR register is set and an interrupt asserted (if enabled).

The address that the error occurred at will be indicated by the addresses stored in the VME and PCI Address registers. After an error on the source bus, the source bus registers will contain the address that the error occurred at. This will be independent of the size or type of transfers being performed on the bus. After an error on the destination bus, the destination address registers indicate a 128-bit window where the error occurred. The errored address will lie at an aligned 64-bit address somewhere before the stored address, or at an aligned 64-bit address contained in the address register. For example, if an error occurs on the destination bus and the address in the destination registers reads as 0x1000\_1004, then the error can be said to lie somewhere in the range of 0x1000\_0FF8 to 0x1000\_1007.

Independent of the error occurring on the destination or source bus, the DMA\_CTL register contains the attributes relevant to the particular DMA packet. It contains such information as: direction of transfer, address spaces accessed and data width. In addition, the DMA\_TBC register provides the number of bytes remaining to transfer on the PCI side.

To generate an interrupt from a DMA error, there are two bits in the DGCS register (and one bit apiece in the VINT\_EN and LINT\_EN registers). In the DGCS register the INT\_LERR bit enables the DMA to generate an interrupt to the Interrupt Channel encountering an error on the PCI bus. The INT\_VERR enables the DMA to generate an interrupt upon encountering an error on the VMEbus. Upon reaching the Interrupt Channel, all DMA interrupts can be routed to either the PCI bus or VME bus by setting the appropriate bit in the enable registers. All DMA sources of interrupts (Done, Stopped, Halted, VME Error, and PCI Error) constitute a single interrupt into the Interrupt Channel.

#### 2.5.2.4 Parity Errors

The Universe both monitors and generates parity information on the PAR# signal. The Universe monitors PAR# when it accepts data as a master during a read or a slave during a write. The Universe drives PAR# when it provides data as a slave during a read or a master during a write. The Universe also drives PAR# during the address phase of a transaction when it is a master and monitors PAR# during an address phase when it is the PCI slave. In both address and data phases, the PAR# signal provides even parity for C/BE#[3:0] and AD[31:0]. If the Universe is powered up in a 64-bit PCI environment, then PAR64 provides even parity for C/BE#[7:4] and AD[63:32].

The PERESP and SERR\_EN bits in the PCI\_CS register (Table A.3) determine whether or not the Universe responds to parity errors. Data parity errors are reported through the assertion of PERR# if the PERESP bit is set. Address parity errors, reported through the SERR# signal, are reported if both PERESP and SERR\_EN are set. Regardless of the setting of these two bits, the D\_PE (Detected Parity Error) bit in the PCI\_CS register is set if the Universe encounters a parity error as a master or as a slave. The DP\_D (Data Parity Detected) bit in the same register is only set if parity checking is enabled through the PERESP bit and the Universe detects a parity error while it is PCI master (i.e. it asserts PERR# during a read transaction or receives PERR# during a write).

No interrupts are generated by the Universe either as a master or as a slave in response to parity errors reported during a transaction. Parity errors are reported by the Universe through assertion of PERR# and by setting the appropriate bits in the PCI\_CS register. If PERR# is asserted to the Universe while it is PCI master, the only action it takes is to set the DP\_D. Regardless of whether the Universe is the master or slave of the transaction, and regardless which agent asserted PERR#, the Universe does not take any action other than to set bits in the PCI\_CS register. The Universe continues with a transaction independent of any parity errors reported during the transaction.

Similarly, address parity errors are reported by the Universe (if the SERR\_EN bit is set) by asserting the SERR# signal and setting the S\_SERR (Signalled SERR#) bit in the PCI\_CS register. Assertion of SERR# can be disabled by clearing the SERR\_EN bit in the PCI\_CS register. No interrupt is generated, and regardless of whether assertion of SERR# is enabled or not, the Universe does not respond to the access with DEVSEL#. Typically the master of the transaction times out with a master-abort. As a master, the Universe does not monitor SERR#. It is expected that a central resource on the PCI bus will monitor SERR# and take appropriate action.

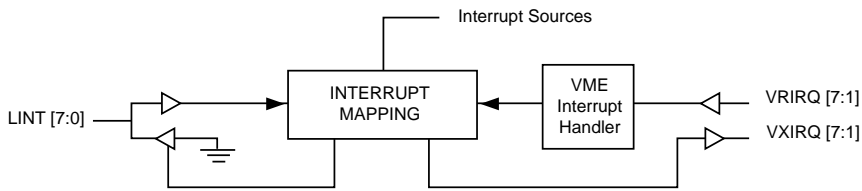
## 2.6 Interrupter

### 2.6.1 The Interrupt Channel

The Universe Interrupt Channel is a module within the Universe which handles prioritization and routing of various interrupt sources to interrupt outputs on the PCI bus and VMEbus. The interrupt sources are:

- the PCI LINT[7:0] lines,
- the VME IRQ\*[7:1] lines,
- ACFAIL\* and SYSFAIL\*
- various internal signals

These sources can be routed to either the PCI LINT [7:0] lines or the VMEbus IRQ\* [7:0] lines (see Figure 2.10 below). Each interrupt source is individually maskable and can be mapped to various interrupt outputs. Most interrupt sources can be mapped to one particular destination bus. The PCI sources, LINT[7:0], can only be mapped to the VMEbus interrupt outputs, while the VMEbus sources, VIRQ[7:1], can only be mapped to the PCI interrupt outputs. Some internal sources (for example, error conditions or DMA activity) can be mapped to either bus.



**Figure 2.10 : Interrupt Sources and Destinations**

## 2.6.2 PCI Interrupt Generation

The Universe expands on the basic PCI specification which permits “single function” devices to assert only a single interrupt line. Eight PCI interrupt outputs are provided to give maximum flexibility, although if full PCI compliancy is required, the user may route all interrupt sources to a single PCI interrupt output.



*NOTE: Only one of the PCI interrupt outputs, LINT[0], has the drive strength to be fully compliant with the PCI specification. The other seven may require buffering if they are to be routed to PCI compliant interrupt lines. For most applications, however, the drive strength provided should be sufficient.*

PCI interrupts may be generated from multiple sources:

- VME interrupts
  - IRQ\*[7:1]
  - SYSFAIL\* (SYSFAIL bit)
  - ACFAIL\*

- internal interrupts
  - DMA
  - VME bus error encountered
  - PCI bus error encountered
  - VMEbus ownership has been granted while VOWN bit is set (see “Exclusive Accesses” on page 2-41)
  - software interrupt
  - VME IACK cycle performed in response to a software interrupt

Each of these sources may be individually enabled in the LINT\_EN register (Table A.40) and mapped to a single LINT signal through the LINT\_MAP0 and LINT\_MAP1 registers (Table A.42 and Table A.43). When an interrupt is received on any of the enabled sources, the Universe asserts the appropriate LINT pin and sets a matching bit in the LINT\_STAT register (Table A.41). See Table 2.14 below for a list of the enable, mapping and status bits for PCI interrupt sources.

**Table 2.13 : PCI Interrupt Source Enabling, Mapping, and Status Bits**

Interrupt Source	Enable Bit in LINT_EN	Mapping Bit in LINT_MAP0 or LINT_MAP1	Status Bit in LINT_STAT
ACFAIL*	ACFAIL	ACFAIL	ACFAIL
SYSFAIL*	SYSFAIL	SYSFAIL	SYSFAIL
PCI Software Interrupt	SW_INT	SW_INT	SW_INT
VME Software IACK	SW_IACK	SW_IACK	SW_IACK
VMEbus Error	VERR	VERR	VERR
PCI Bus Error	LERR	LERR	LERR
DMA Event	DMA	DMA	DMA
VMEbus Interrupt Input	VIRQ7-1	VIRQ7-1	VIRQ7-1
VMEbus Ownership	VOWN	sVOWN	VOWN

The LINT\_STAT register shows the status of all sources of PCI interrupts, independent of whether that source has been enabled. This implies that an interrupt handling routine must mask out those bits in the register that do not correspond to enabled sources on the active LINT pin.



Except for SYSFAIL\* and ACFAIL\*, all sources of PCI interrupts are edge sensitive. Enabling of the ACFAIL\* or SYSFAIL\* sources (ACFAIL and SYSFAIL bits in the LINT\_EN register) cause the status bit and mapped PCI interrupt pin to assert synchronously with the assertion of the ACFAIL\* or SYSFAIL\* source. The PCI interrupt is negated once the ACFAIL\* or SYSFAIL\* source is also negated and its respective status bit in the LINT\_STAT register is cleared, or the interrupt is disabled. Both of these sources are synchronized and filtered by the 64 MHz clock at their inputs.

All other sources of PCI interrupts are edge sensitive. Note that the VMEbus source for PCI interrupts actually comes out of the VMEbus Interrupt Handler block and reflects acquisition of a VME STATUS/ID. Therefore, even though VMEbus interrupts externally are level sensitive as required by the VME specification, they are internally mapped to edge-sensitive interrupts (see “VMEbus Interrupt Handling” on page 2-61).

The interrupt source status bit (in the LINT\_STAT register) and the mapped LINT pin remain asserted with all interrupts. The status bit and the PCI interrupt output pin are only released when the interrupt is cleared by writing a “one” to the appropriate status bit.

### 2.6.3 VMEbus Interrupt Generation

Interrupts may be generated on any combination of VME interrupts from multiple sources:

- PCI Interrupts
  - LINT[7:0]
- Internal Interrupts
  - DMA
  - VME bus error encountered
  - PCI bus error encountered
  - software interrupt

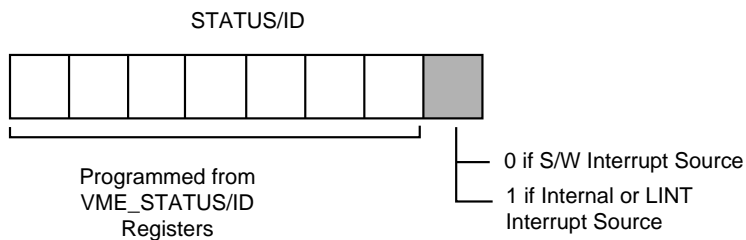
Each of these sources may be individually enabled through the VINT\_EN register (Table A.44) and mapped to a particular VME Interrupt level using the VINT\_MAP0 and VINT\_MAP1 registers (Table A.46 and Table A.47). Multiple sources may be mapped to any VME level. Mapping interrupt sources to level 0 effectively disables the interrupt.

Once an interrupt has been received from any of the sources, the Universe sets the corresponding status bit in the VINT\_STAT register (Table A.45), and asserts the appropriate VMEbus interrupt output signal (if enabled). When a VME interrupt handler receives the interrupt, it will perform an IACK cycle at that level. When the Universe decodes that IACK cycle together with IACKIN\* asserted, it provides the STATUS/ID previously stored in the STATID register (Table A.48). See Table 2.14 below for a list of the enable, mapping and status bits for VMEbus interrupt sources.

**Table 2.14 : VMEbus Interrupt Source Enabling, Mapping, and Status Bits**

Interrupt Source	Enable Bit in VINT_EN	Mapping Bit in VINT_MAP0 or VINT_MAP1	Status Bit in VINT_STAT
VME Software Interrupt	SW_INT	SW_INT	SW_INT
VMEbus Error	VERR	VERR	VERR
PCI Bus Error	LERR	LERR	LERR
DMA Event	DMA	DMA	DMA
PCI bus Interrupt Input	LINT7-0	LINT7-0	LINT7-0

For all VMEbus interrupts, the Universe interrupter supplies a pre-programmed 8-bit STATUS/ID: a common value for all interrupt levels. The upper seven bits are programmed in the STATID register. The lowest bit is cleared if the source of the interrupt was the software interrupt, and is set for all other interrupt sources. If a software interrupt source and another interrupt source are active and mapped to the same VMEbus interrupt level, the Universe gives priority to the software source.



**Figure 2.11 : STATUS/ID Provided by Universe**

Once the Universe has provided the STATUS/ID to an interrupt handler during a software initiated VMEbus interrupt, it generates an internal interrupt, SW\_IACK. If enabled, this interrupt feeds back to the PCI bus (through one of the LINT pins) to signal a process that the interrupt started through software has been completed.

All VMEbus interrupts generated by the Universe are RORA, except for the software interrupt which is ROAK. This means that if the interrupt source was the software interrupt, then the VMEbus interrupt output is automatically negated when the Universe receives the IACK cycle. However, for any other interrupt, the VMEbus interrupt output remains asserted until cleared by a register access. Writing a “one” to the relevant bit in the VINT\_STAT register clears that interrupt source. However, since PCI interrupts are level sensitive, if an attempt is made to clear the VMEbus interrupt while the LINT pin is still asserted, the VMEbus interrupt remains asserted. This causes a second interrupt to be generated to the VMEbus. For this reason, a VME interrupt handler should clear the source of the PCI interrupt before clearing the VMEbus interrupt.

## 2.7 Interrupt Handling

### 2.7.1 PCI Interrupt Handling

All eight PCI interrupt lines, LINT[7:0], can act as interrupt inputs to the Universe. They are level sensitive and, if enabled in the VINT\_EN register (Table A.44), immediately generate an interrupt to the VMEbus. It is expected that when a VMEbus interrupt handler receives the Universe’s STATUS/ID from the Universe, the interrupt handler will clear the VMEbus interrupt by first clearing the source of the interrupt on the PCI bus, and then clearing the VMEbus interrupt itself (by writing a “one” to the appropriate bit in the VINT\_STAT register, Table A.45).

Note that since PCI interrupts are level sensitive, if an attempt is made to clear the VMEbus interrupt while the LINT pin is still asserted, the VMEbus interrupt remains asserted. This causes a second interrupt to be generated to the VMEbus. For this reason, a VMEbus interrupt handler should clear the source of the PCI interrupt before clearing the VMEbus interrupt.

### 2.7.2 VMEbus Interrupt Handling

As a VME interrupt handler, the Universe can monitor any or all of the VMEbus interrupt levels. It can also monitor SYSFAIL\* and ACFAIL\*, although IACK cycles are not generated for these inputs. Each interrupt is enabled through the LINT\_EN register (Table A.40).

Once enabled, assertion of any of the VMEbus interrupt levels, IRQ[7:1]\*, causes the internal interrupt handler circuitry to request ownership of the Universe's VME Master Interface on the level programmed in the MAST\_CTL register (see “VMEbus Requester” on page 2-6). This interface is shared between several channels in the Universe: the PCI Slave Channel, the DMA Channel, and the Interrupt Channel. The Interrupt Channel has the highest priority over all other channels and, if an interrupt is pending, assumes ownership of the VME Master Interface when the previous owner has relinquished ownership.

There may be some latency between reception of a VMEbus interrupt and generation of the IACK cycle. This arises because of the latency involved in the Interrupt Channel gaining control of the VME master interface, and because of possible latency in gaining ownership of the VMEbus if the VME Master Interface is programmed for release-when-done. In addition, the Universe only generates an interrupt on the PCI bus once the IACK cycle has completed on the VMEbus. Because of these combined latencies (time to acquire VMEbus and time to run the IACK cycle), systems should be designed to accommodate a certain worst case lag time from VMEbus interrupt generation to its translation to the PCI bus.

When the Universe receives a STATUS/ID in response to an IACK cycle, it stores that value in one of seven registers. These registers, V1\_STATID through V7\_STATID (Table A.49 to Table A.55), store the STATUS/ID corresponding to each IACK level (in the STATID field). Once an IACK cycle has been generated and the resulting STATUS/ID is latched, another IACK cycle will not be run on that level until the level has been re-armed by writing a “one” to the corresponding status bit in the VINT\_STAT register (Table A.45). If other interrupts (at different levels) are pending while the interrupt is waiting to be re-armed, IACK cycles are run on those levels in order of priority and the STATUS/IDs stored in their respective registers.

Once the IACK cycle is complete and the STATUS/ID stored, an interrupt is generated to the PCI bus on one of LINT#[7:0] interrupt output lines depending on the mapping for that VME level in the LINT\_MAP0 register. The interrupt is cleared and the VMEbus interrupt level is re-armed by clearing the correct bit in the LINT\_STAT register.

### 2.7.2.1 Bus Error During VME IACK Cycle

A bus error encountered on the VMEbus while the Universe is performing an IACK cycle is handled by the Universe in two ways. The first is through the error logs in the VME Master Interface. These logs store address and command information whenever the Universe encounters a bus error on the VMEbus (see “Bus Error Handling” on page 2-52). If the error occurs during an IACK cycle, the IACK# bit is set in the V\_AMERR register (Table A.80). The VME Master Interface also generates an internal interrupt to the Interrupt Channel indicating a VMEbus error occurred. This internal interrupt can be enabled and mapped to either the VMEbus or PCI bus.

As well as generating an interrupt indicating an error during the IACK cycle, the Universe also generates an interrupt as though the IACK cycle completed successfully. If an error occurs during the fetching of the STATUS/ID, the Universe sets the ERR bit in the Vn\_STATID register (Table A.49 to Table A.55), and generates an interrupt on the appropriate LINT pin (as mapped in the LINT\_MAP0 register, Table A.42). The PCI resource, upon receiving the PCI interrupt, is expected to read the STATUS/ID register, and take appropriate actions if the ERR bit is set. Note that the STATUS/ID cannot be considered valid if the ERR bit is set in the STATUS/ID register.

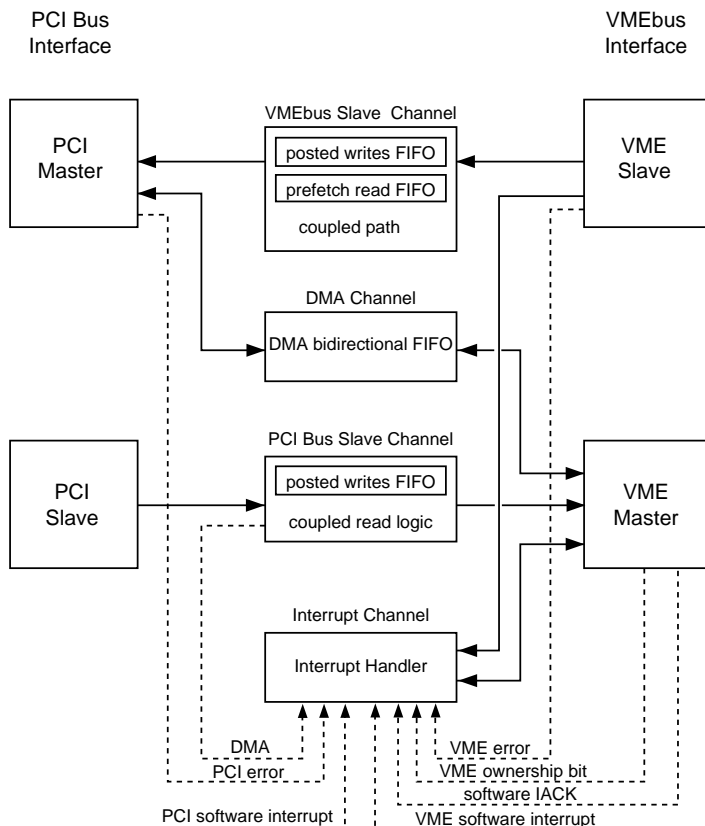
It is important to recognize that the IACK cycle error may generate two PCI interrupts: one through the VMEbus master bus error interrupt and another through the standard PCI interrupt translation. Should an error occur during acquisition of a STATUS/ID, the VINT\_STAT register (Table A.45) will show that both VIRQx, and VERR are active.

### 2.7.3 Internal Interrupt Handling

The Universe's internal interrupts are routed from several processes in the device (see Figure 2.12 on page 2-64). There is an interrupt from the VME master interface to indicate a VMEbus error, another from the PCI master interface to indicate an error on that bus, another from the DMA to indicate various conditions in that channel, along with several others as indicated in Table 2.15 below. Some interrupts may be routed only to the PCI bus, some only to the VMEbus, and others that may be routed to both (although we recommend that you map interrupts to a single bus).

**Table 2.15 : Internal Interrupt Routing**

Interrupt Source	May be Routed to:	
	VMEbus	PCI Bus
PCI bus errors	√	√
VME bus error	√	√
DMA event	√	√
IACK cycle complete for s/w interrupt		√
VME bus ownership granted		√
PCI s/w interrupt		√
VME s/w interrupt	√	



**Figure 2.12 : Universe Sources of Internal Interrupts**

### 2.7.3.1 VME and PCI Software Interrupts

Two interrupt sources are provided as mechanisms to interrupt the PCI or VME buses through software. These interrupts may be triggered by writing a “one” to the respective enable bits. Writing a “one” to the SW\_INT bit in the VINT\_EN register (Table A.44) triggers one (and only one) interrupt on the VMEbus on the level programmed in the VINT\_MAP1 register (Table A.47). When an IACK cycle is serviced on the VMEbus, the Universe can be programmed to generate an interrupt on the PCI bus by setting the SW\_IACK enable bit in the LINT\_EN register. Writing a “one” to the SW\_INT in the LINT\_EN (Table A.40) register

similarly generates an interrupt to the PCI bus. While these interrupt sources are active, their respective status bits are set (SW\_INT in VINT\_STAT; SW\_IACK and SW\_INT in LINT\_STAT). Either interrupt may be cleared by clearing the respective enable bit although clearing a VMEbus interrupt through this method is not recommended since it may result in a spurious interrupt on that bus.

Since the software interrupt is edge-sensitive, the SW\_INT bit should be cleared before generating another interrupt. It is recommended that the appropriate interrupt handler clear this bit once it has completed its operations. Alternatively, the process generating the interrupt could clear this bit before re-asserting it.

Software interrupts on the VMEbus have priority over other interrupts mapped internally to the same level on the VMEbus. When a VMEbus interrupt handler generates an IACK cycle on a level mapped to both a software interrupt and another interrupt, the Universe always provides the STATUS/ID for the software interrupt (bit zero of the Status/ID is cleared). If there are no other active interrupts on that level, the interrupt is automatically cleared upon completion of the IACK cycle (since software interrupts are ROAK).

While the software interrupt STATUS/ID has priority over other interrupt sources, the user can give other interrupt sources priority over the software interrupt. This is done by reading the LINT\_STAT register (Table A.41) when handling a Universe interrupt. This register indicates all active interrupt sources. Using this information, the interrupt handler can then handle the interrupt sources in any system-defined order.

### 2.7.3.2 Software IACK Interrupt

The Universe generates an internal interrupt when it provides the software STATUS/ID to the VMEbus. This interrupt can only be routed to a PCI interrupt output. A PCI interrupt will be generated upon completion of an IACK cycle that had been initiated by the Universe's software interrupt if the SW\_IACK bit in the LINT\_EN register (Table A.40) is set and the SW\_IACK field in the LINT\_MAP1 register (Table A.43) is mapped to a corresponding PCI interrupt line. This interrupt could be used by a PCI process to indicate that the software interrupt generated to the VMEbus has been received by the slave device and acknowledged.

Like other interrupt sources, this interrupt source can be independently enabled through the LINT\_EN register (Table A.40) and mapped to a particular LINT pin using the LINT\_MAP1 register (Table A.43). A status bit in the LINT\_STAT register (Table A.41) indicates when the interrupt source is active, and is used to clear the interrupt once it has been serviced.

### 2.7.3.3 VME Ownership Interrupt

The VMEbus ownership interrupt is generated when the Universe acquires the VMEbus in response to programming of the VOWN bit in the MAST\_CTL register (Table A.56). This interrupt source can be used to indicate that ownership of the VMEbus is ensured during an exclusive access (see "Exclusive Accesses" on page 2-41). The interrupt is cleared by writing a one to the matching bit in the LINT\_STAT register (Table A.41).

### 2.7.3.4 DMA Interrupt

The DMA module provides six possible interrupt sources:

- if the DMA is stopped (INT\_STOP),
- if the DMA is halted (INT\_HALT),
- if the DMA is done (INT\_DONE),
- for PCI bus errors (INT\_LERR),
- for VMEbus errors (INT\_VERR), or
- if there is a PCI protocol error or if the Universe is not enabled as PCI master (INT\_M\_ERR).

All of these interrupt sources are ORed to a single DMA interrupt output line. When an interrupt comes from the DMA module, software must read the DMA status bits (Table A.38) to discover the originating interrupt source. The DMA interrupt can be mapped to either the VMEbus or one of the PCI interrupt output lines.

### 2.7.3.5 PCI and VMEbus Error Interrupts

Interrupts from bus errors (either PCI or VMEbus) are generated only when bus errors arise during decoupled writes. The bus error interrupt (from either a PCI or VMEbus error) can be mapped to either a VMEbus or PCI interrupt output line.

## 2.7.4 VME64 Auto-ID

The Universe includes a power-up option for participation in the VME64 Auto-ID process. When this option is enabled, the Universe generates a level 2 interrupt on the VMEbus before release of SYSFAIL\*. When the level 2 IACK cycle is run by the system Monarch, the Universe responds with the Auto-ID Status/ID, 0xFE, and enables access to a CR/CSR image at base address 0x00\_0000.

When the Monarch detects an Auto-ID STATUS/ID on level 2, it is expected to access the enabled CR/CSR space of the interrupter. From there it completes identification and configuration of the card. The Monarch functionality is typically implemented in software on one card in the VMEbus system.



## 2.8 DMA Controller

The Universe uses a DMA controller for high performance data transfer between the PCI and VME buses. It is operated through a series of registers that control the source and destination for the data, length of the transfer and the transfer protocol to be used. There are two modes of operation for the DMA: Direct Mode, and Linked List Mode. In direct mode, the DMA registers are programmed directly by the software. In linked list mode, the registers are automatically loaded from PCI memory by the Universe, and the transfer described by these registers is executed. The block of DMA registers stored in PCI memory is called a command packet. A command packet may be linked to another command packet, such that when the DMA has completed the operations described by one command packet, it automatically moves on to the next command packed in the linked-list of command packets.

This section is broken into six major sub-sections:

1. “DMA Registers Outline” on page 2-67 describes in detail how the DMA is programmed from a register perspective.
2. “Direct Mode Operation” on page 2-73 describes how to operate the DMA when directly programming the DMA registers.
3. “Linked-List Operation” on page 2-76 describes how to operate the DMA when a linked-list of command packets describing DMA transfers is stored in PCI memory.
4. “FIFO Operation and Bus Ownership” on page 2-82 describes internally how the DMA makes use of its FIFO and how this affects ownership of the VME and PCI buses.
5. “DMA Interrupts” on page 2-85 describes the interrupts generated by the DMA
6. “DMA Error Handling” on page 2-86 describes how to handle errors encountered by the DMA

### 2.8.1 DMA Registers Outline

The DMA registers reside in a block starting at offset 0x200. Loosely speaking, they describe a single DMA transfer: where to transfer data from; where to transfer data to; how much data to transfer; and what data mode on the PCI and VME buses to use to transfer the data. A final register contains status and control information for the transfer. While the DMA is active, the registers are locked against any changes so that any writes to the registers will have no impact.

In direct-mode operation, these registers would be programmed directly by the user. In linked-list operation, they are repeatedly loaded by the Universe from command packets residing in PCI memory until the end of the linked-list is reached (see “Linked-List Operation” on page 2-76).

### 2.8.1.1 Source and Destination Addresses

The source and destination addresses for the DMA reside in two registers: the DMA PCI Bus Address Register (DLA register, Table A.35), and the DMA VMEbus Address Register (DVA register in Table A.36). The determination of which is the source address, and which is the destination is made by the L2V bit in the DCTL register (Table A.33). When set, the DMA transfers data from the PCI to the VMEbus. Hence DLA becomes the PCI source register and DVA becomes the VME destination register. When cleared, the DMA transfers data from the VME to PCI bus and DLA becomes the PCI destination register; DVA becomes the VME source register.

The PCI address may be programmed to any byte address in PCI memory space. It can not transfer to or from PCI IO or Configuration spaces.

The VME address may also be programmed to any byte address, and can access any VME address space from A16 to A32 in supervisory or non-privileged space, and data or program space. The setting of address space, A16, A24 or A32, is programmed in the VAS field of the DCTL register (Table A.33). The sub-spaces are programmed in the PGM and SUPER fields of the same register.



*Note : Although the PCI and VME addresses may be programmed to any byte aligned address, they must be 8-byte aligned to each other (i.e.. the low three bits of each must be identical). If not programmed with aligned source and destination addresses and the DMA is started, the DMA will immediately stop, set the protocol error bit (P\_ERR) in the DCSR register (Table A.38), and if enabled to, generate an interrupt. Linked-list operations will cease.*

The source and destination address registers are updated continuously by the DMA to allow the user to monitor DMA progress. If read during DMA activity, the DLA will return the address of the next PCI transfer, and the DTBC will return the number of bytes remaining to transfer on the PCI side. The DVA register may be incorrect from the actual active VME address by the contents of the FIFO. All of the DMA registers are locked against any changes by the user while the DMA is active. When stopped due to an error situation, these registers give an indication of where the error was encountered. See “DMA Error Handling” on page 2-86 for details. At the end of a transfer, these the DVA and DLA registers will point to the next address at the end of the transfer block, and the DTBC register will be zero.

### 2.8.1.2 Transfer Size

The DMA may be programmed through the DMA Transfer Byte Count register (DTBC register in Table A.34) to transfer any number of bytes from 1 byte to 16 MBytes. There are no alignment requirements to the source or destination addresses. Should the width of the data turnovers (8- through 64-bit on VME and 32- or 64-bit on PCI) not align to the length of the transfer or the source/destination addresses, the DMA will insert transfers of smaller width on the appropriate bus. For example, if a 15-byte transfer is programmed to start at address 0x1000 on the VME bus, and the width is set for D32, the DMA will perform three D32 transfers, followed by a D16 transfer, followed by a D08 transfer. The Universe does not generate unaligned transfers. On a 32-bit PCI bus, if the start address was 0x2000, the DMA would generate three data beats with all byte lanes enabled, and a fourth with three byte lanes enabled.

The DTBC register is not updated while the DMA is active (indicated by the ACT bit in the DGCS register). At the end of a transfer it will contain zero. If stopped by the user (via the STOP bit in the DGCS register) or the DMA encounters an error, the DTBC register contains the number of bytes remaining to transfer on the source side. See “DMA Error Handling” on page 2-86.

Starting the DMA while DTBC=0, will result in one of two situations. If the CHAIN bit in the DGCS register (Table A.38) is not set, the DMA will not start; it will perform no action. If the CHAIN bit is set, then the DMA loads the DMA registers with the contents of the command packet pointed to by the DCPPI register (Table A.37), and starts the transfers described by that packet.

### 2.8.1.3 Transfer Data Width

The VME bus and PCI bus data widths are determined by three fields in the DCTL register (Table A.33). These fields affect the speed of the transfer. They should be set for the maximum allowable width that the targeted destination device is capable of accepting.

On the VMEbus, the DMA supports the following data widths:

D08(E), D16, D32, D64, D32BLT, and D64BLT (MBLT).

The DMA does not support D08(E), D08(O), and D16BLT. The width of the transfer is set with the VDW field in the DCTL register. Block transfers are enabled by setting the VCT bit in the same register.



*Caution: The DMA may generate illegal VME cycles if programmed for D08 or D16 and block transfers are enabled.*

The Universe may perform data transfers smaller than that programmed in the VDW field in order to bring itself into alignment with the programmed width. For example if the width is set for D32 and the starting VME address is 0x101, the DMA will perform a D08 cycle followed by a D16 cycle. Only once it has achieved the alignment set in the VDW field does it start D32 transfers. At the end of the transfer, the DMA will also have to perform more low-width transfers if the last address is not aligned to VDW. Similarly, if the VCT bit is set to enable block transfers, the DMA may perform non-block transfers to bring itself into alignment.

On the PCI bus, the DMA provides the option of performing 32- or 64-bit PCI transactions through the LD64EN bit in the DCTL register (Table A.33). If the Universe has powered-up on a 32-bit bus (see “Power-up Option Descriptions” on page 2-99), this bit will have no effect. If powered-up on a 64-bit bus, this bit can provide some performance improvements when accessing 32-bit targets on that bus. Following the PCI specification, before a 64-bit PCI initiator starts a 64-bit transaction, it engages in a protocol with the intended target to determine if it is 64-bit capable. This protocol typically consumes one clock period. To save bandwidth, the LD64EN bit can be cleared to bypass this protocol when it is known that the target is only 32-bit capable

### **2.8.1.4 DMA Command Packet Pointer**

The DMA Command Packet Pointer (DCPP in Table A.37) points to a 64-bit aligned address location in PCI memory space that contains the next command packet to be loaded once the transfer currently programmed into the DMA registers has been successfully completed. When it has been completed (or the DTBC register is zero when the GO bit is set) the DMA reads the 32 byte command packet from PCI memory and executes the transfer it describes.

### **2.8.1.5 DMA Control and Status**

The DMA General Control/Status Register (DGCS in Table A.38) contains a number of fields that control initiation and operation of the DMA as well as actions to be taken on completion.

#### **DMA Initiation**

Once all the parameters associated with the transfer have been programmed (source/destination addresses, transfer length and data widths, and if desired, linked lists enabled), the DMA transfer is started by setting the GO bit in the DGCS register. This causes the DMA to first examine the DTBC register. If it is non-zero, it latches the values programmed into the DCTL, DTBC, DLA, and DVA registers and initiates the transfer programmed into those registers. If DTBC=0, it checks the CHAIN bit in the DGCS register and if that bit is cleared it assumes the transfer to have completed and stops. Otherwise, if the CHAIN bit is set, it loads into the DMA registers the command packet pointed to by the DCPP register and initiates the transfer describe there.

If the GO bit is set, but the Universe has not been enabled as a PCI master with the BM (bus master enable) bit in the PCI\_CSR register, or if the DVA and DLA contents are not 64-bit aligned to each other, the transfer does not start, a protocol error is indicated in the P\_ERR bit in the DGCS register and, if enabled, an interrupt is generated.

If the DMA has been terminated for some reason (stopped, halted, or error), all DMA registers contain values indicating where the DMA terminated. Once all status bits have been cleared, the DMA may be restarted from where it left off by simply setting the GO bit. The GO bit will only have an effect if all status bits have been cleared. These bits include STOP, HALT, DONE, LERR, VERR, and P\_ERR; all in the DGCS register. These bits are all cleared by writing "one" to them, either before or while setting the GO bit.

The GO bit always return a zero when read independent of what state the DMA is in. Clearing the bit has no impact at any time. The ACT bit in the DGCS register indicates whether the DMA is currently active. It is set by the DMA once the GO bit is set, and cleared when the DMA is idle. Generally, when the ACT bit is cleared, one of the other status bits in the DGCS register is set (DONE, STOP, HALT, LERR, VERR, or P\_ERR), indicating why the DMA is no longer active.

### DMA VMEbus Ownership

Two fields in the DGCS register determine how the DMA will share the VMEbus with the other two potential masters in the Universe (PCI slave channel, and Interrupt channel), and with other VME masters on the bus. These fields are:

- VON, which affects how much data the DMA will transfer before giving the opportunity to another master (either internal or external) to assume ownership of the bus.
- VOFF, which affects how long the DMA remains off the bus after the VON limit has been reached before the DMA will re-request it.

VON may be disabled by setting the field to zero. When set as such, the DMA will continue transferring data as long as it is able. With non-zero settings, the DMA will relinquish ownership of the bus when it reaches defined boundaries: 256, 512, 1K, 2k, 4K, 8K, or 16Kbytes. There are other conditions under which the DMA may relinquish bus ownership. See "FIFO Operation and Bus Ownership" on page 2-82 for details on the VMEbus request and release conditions for the DMA.

By setting VOFF to zero, the DMA will immediately re-request the bus once the VON boundary has been reached. Since the DMA operates in a round-robin fashion with the PCI slave channel, and a priority fashion with the interrupt channel, if either of these channels require ownership of the VMEbus, they will receive it at this time.

VOFF is only invoked when VMEbus tenure is relinquished due to encountering the VON boundary. When the bus is released due to other conditions (e.g., the DMA FIFO has gone full while reading from the VMEbus), it will be re-requested as soon as that condition is cleared. The VOFF timer can be programmed to various time intervals from 0 $\mu$ s to 1024 $\mu$ s. See “FIFO Operation and Bus Ownership” on page 2-82 for details on the VMEbus request and release conditions for the DMA.

### **DMA Completion and Termination**

Normally, the DMA will continue processing its transfers and command packets until either it completes everything has been requested to, or it encounters an error. There are also two methods for the user to interrupt this process and cause the DMA to terminate prematurely: Stop and Halt. Stop causes the DMA to terminate immediately, while halt causes the DMA to terminate when it has completed processing the current command packet in a linked list.

When the STOP\_REQ bit in the DGCS register is set by the user, it tells the DMA to cease its operations on the source bus immediately. Remaining data in the FIFO continues to be written to the destination bus until the FIFO is empty. Once the FIFO is empty, the STOP bit in the same register is set and, if enabled, an interrupt generated. The DMA register will contain the values that the DMA stopped at: the DTBC contains the number of bytes remaining in the transfer, the source and destination address registers contain the next address to be read/written, the DCPD contains the next command packet in the linked-list, and the DCTL contains the transfer attributes.

If read transactions are occurring on the VMEbus, then setting a stop request can be affected by the VOFF timer. If the STOP\_REQ bit is set while the DMA is lying idle waiting for VOFF to expire before re-commencing reads, then the request remains pending until the VOFF timer has expired and the bus has been granted.

Halt provides a mechanism to interrupt the DMA at command packet boundaries during a linked-list transfer. In contrast, a stop requests the DMA to be interrupted immediately, while halt takes effect only when the current command packet is complete. A halt is requested of the DMA by setting the HALT\_REQ bit in the DGCS register. This causes the DMA to complete the transfers defined by the current contents of the DMA registers and, if the CHAIN bit is set, load in the next command packet. The DMA then terminates, the HALT bit in the DGCS register is set, and, if enabled, an interrupt generated.

After a stop or halt, the DMA can be restarted from the point it left off by setting the GO bit; but before it can be re-started, the STOP bit and HALT bit must both be cleared.

Regardless of how the DMA stops whether normal, bus error or user interrupted, the DMA will indicate in the DGCS register why it stopped. The STOP and HALT bits get set in response to a stop or halt request. The DONE bit gets set when the DMA has successfully completed the DMA transfer, including all entries in the linked-list if operating in that mode. There are also three bits that are set in response to error conditions: LERR in the case of target

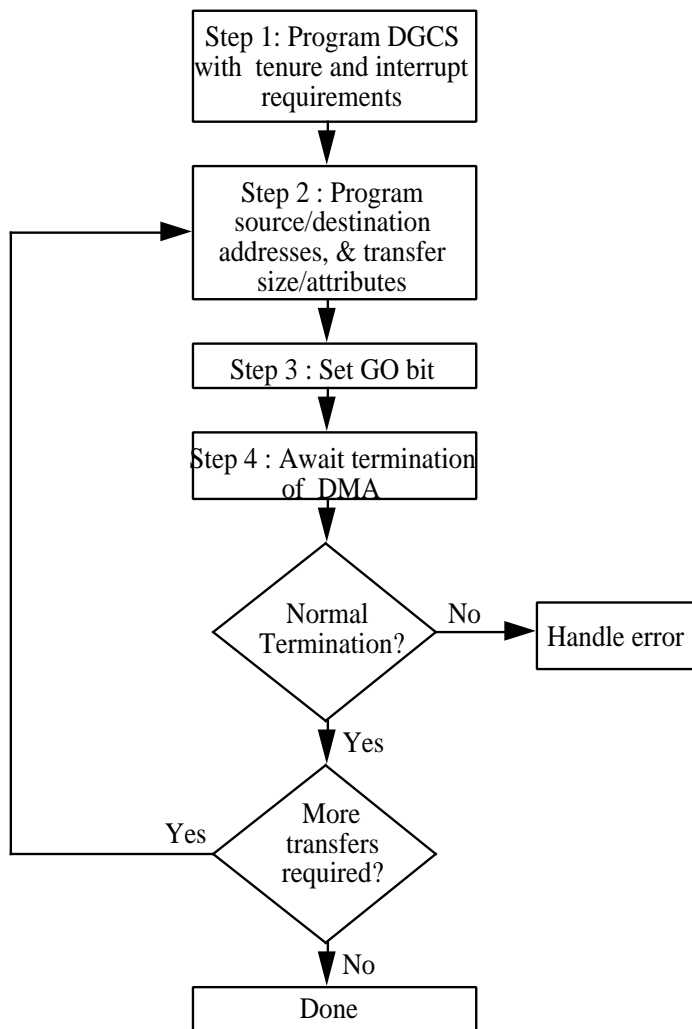
abort encountered on the PCI bus; VERR in the case of a bus error encountered on the VME bus; and P\_ERR in the case that the DMA has not been properly programmed (the DMA was started with the BM bit in the PCI\_CSR register not enabled, or the DLA and DVA registers were not 64-bit aligned, (see “Source and Destination Addresses” on page 2-68). Before the DMA can be restarted, each of these status bits must be cleared.

When the DMA terminates, an interrupt may be generated to VME or PCI buses. The user has control over which DMA termination conditions will cause the interrupt through the INT\_STOP, INT\_HALT, INT\_DONE, INT\_LERR, INT\_VERR, and INT\_P\_ERR bits in the DGCS register.

### 2.8.2 Direct Mode Operation

When operated in direct mode, the Universe DMA is set through manual register programming. Once the transfer described by the DVA, DLA, DTBC and DCTL registers has been completed, the DMA sits idle awaiting the next manual programming of the registers.

Figure 2.13 describes the steps involved in operating the DMA in direct mode.

**Figure 2.13 : Direct Mode DMA transfers**



In Step 1, the DGCS register is set up: the CHAIN bit is cleared, VON and VOFF are programmed with the appropriate values for controlling DMA VMEbus tenure, and the interrupt bits (INT\_STOP, INT\_HALT, INT\_DONE, INT\_LERR, INT\_VERR, and INT\_P\_ERR) are programmed to enable generation of interrupts based on DMA termination events. DMA interrupt enable bits in the LINT\_EN or VINT\_EN bits should also be enabled as necessary (see “PCI Interrupt Generation” on page 2-57 and “VMEbus Interrupt Generation” on page 2-59 for details on generating interrupts).

In Step 2, the actual transfer is programmed into the DMA: source and destination start addresses into the DLA and DVA registers, transfer count into the DTBC register, and transfer width, direction and VME address space into the DCTL register.

In Step 3, with the transfer programmed, the GO bit in DGCS must be set. Ensure that if any status bits (DONE, STOP, HALT, LERR, VERR, or P\_ERR) remain set from a previous transfer they are cleared. This may be done during the same write cycle that sets the GO bit. The ACT bit will be set, and the DMA will then start transferring data, sharing ownership of the VME bus with the PCI slave and Interrupt channels and the PCI bus with the VME slave channel.

In Step 4, one waits for termination of the DMA transfers. The DMA will continue with the transfers until it:

- completes all transfers,
- is terminated early with the STOP\_REQ bit, or
- encounters an error on the PCI or VME buses.

Each of these conditions will cause the ACT bit to clear, and a corresponding status bit to be set in the DGCS register. If enabled in step 1, an interrupt will also be generated. Once the software has set the GO bit, the software can monitor for DMA completion by either waiting for generation of an interrupt, or by polling the status bits. It is recommended that a background timer also be initiated to time-out the transfer. This will ensure that the DMA has not been hung up by a busy VMEbus, or other such system issues.

If an early termination is desired, perhaps because a higher priority operation is required, the STOP\_REQ bit in the DGCS register can be set. This will stop all DMA operations on the source bus immediately, and set the STOP bit in the same register when the last piece of queued data in the DMA FIFO has been written to the destination bus. Attempting to terminate the transfer with the HALT\_REQ bit will have no effect in direct mode operation since this bit only requests the DMA to stop between command packets in linked-list mode operation.

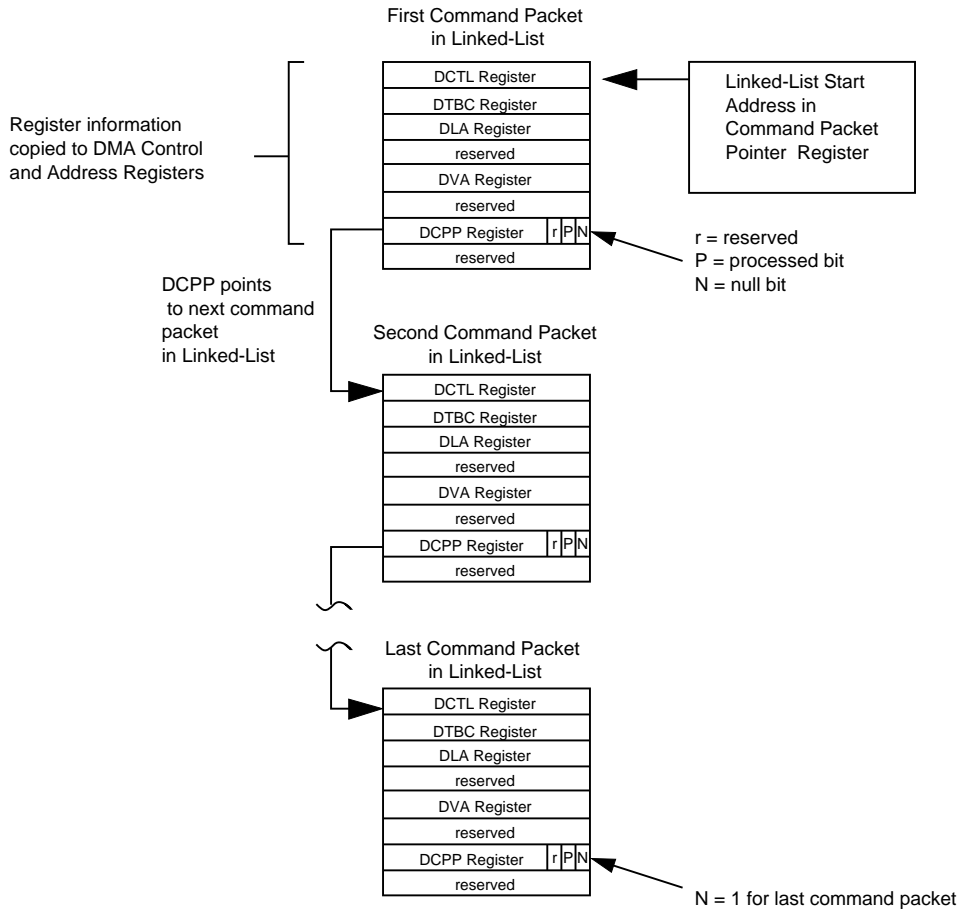
When the software has detected completion, it should check the status bits in the DGCS register to see the reason for completion. If one of the error bits have been set it proceeds into an error handling routine (see “DMA Error Handling” on page 2-86). If the STOP bit was set, the software should take whatever actions were desired when it set the STOP\_REQ bit. For example, if it was stopped for a higher priority transfer, it might record the DLA, DVA and DTBC registers, and then reprogram them with the higher priority transfer. When that has completed it can restore the DVA, DLA and DTBC registers to complete the remaining transfers.

If the DONE bit was set, it indicates that the DMA completed its requested transfer successfully, and if more transfers are required, the software can proceed to step 2 to start a new transfer.

### 2.8.3 Linked-List Operation

Unlike direct mode, in which the DMA performs a single block of data at a time, linked-list mode allows the DMA to transfer a series of non-contiguous blocks of data without software intervention. Each entry in the linked-list is described by a command packet which parallels the DMA register layout. The data structure for each command packet is the same (see Figure 2.14 below), and contains all the necessary information to program the DMA address and control registers. It could be described in software as a record of eight 32-bit data elements. Four of the elements represent the four core registers required to define a DMA transfer: DCTL, DTBC, DVA, and DLA. A fifth element represents the DCP register which points to the next command packet in the list. The least two significant bits of the DCP element (the processed and null bits) provide status and control information for linked list processing.

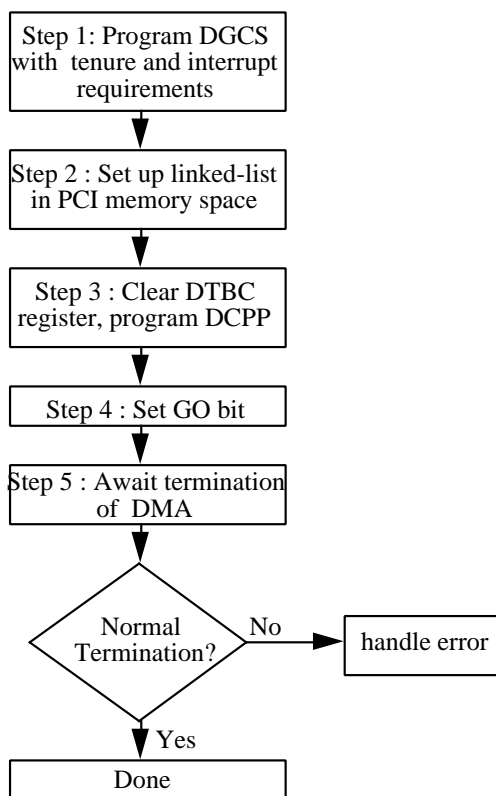
The processed bit indicates whether a command packet has been processed or not. When the DMA processes the command packet and has successfully completed all transfers described by this packet, it sets the processed bit to “1” before reading in the next command packet in the list. This implies that the processed bit must be initially set for “0” by the user for it to be of use. This bit, when set to 1, indicates that this command packet has been disposed of by the DMA and its memory can be de-allocated or reused for another transfer description.



**Figure 2.14 : Command Packet Structure and Linked List Operation**

The null bit indicates the termination of the entire linked list. If the null bit is set to “0”, the DMA processes the next command packet pointed to by the command packet pointer. If the null bit is set to “1” then the address in the command packet pointer is considered invalid and the DMA stops at the completion of the transfer described by the current command packet.

Figure 2.14 outlines the steps in programming the DMA for linked-list operation. In step 1, the DGCS register is set up: the CHAIN bit is set, VON and VOFF are programmed with the appropriate values for controlling DMA VMEbus tenure, and the interrupt bits (INT\_STOP, INT\_HALT, INT\_DONE, INT\_LERR, INT\_VERR, and INT\_P\_ERR) are programmed to enable generation of interrupts based on DMA termination events. DMA interrupt enable bits in the LINT\_EN or VINT\_EN bits should also be enabled as necessary (“PCI Interrupt Generation” on page 2-57 and “VMEbus Interrupt Generation” on page 2-59).



**Figure 2.15 : DMA Linked List Operation**

Next the linked-list structure is programmed with the required transfers. The actual structure may be set up at any time with command packet pointers pre-programmed and then only the remaining DMA transfer elements need be programmed later. One common way is to set up the command packets as a circular queue: each packet points to the next in the list, and the last points to the first. This allows continuous programming of the packets without having to set-up or tear down packets later.

Once the structure for the linked-list is established, the individual packets are programmed with the appropriate source and destination addresses, transfer sizes and attributes.

To start the linked-list, clear the DTBC register, program the DCPPI register to point to the first command packet in the list, and then set the GO bit in the DGCS register. If the DTBC register is non-zero when the GO bit is set, the DMA will first perform the transfers defined by the current contents of the DCTL, DTBC, DVA and DLA registers. Once that is complete it will then start the transfers defined by the linked-list pointed to in the DCPPI register.

Once the DMA channel is enabled, it processes the first command packet as specified by the DCPPI register. The DMA transfer registers are programmed by information in the command packets and the DMA transfer steps along each command packet in sequence (see Figure 2.14 above). The DMA will terminate when it:

- processes a command packet with the null bit set indicating the last packet of the list
- is stopped with the STOP\_REQ bit in the DGCS register
- is halted with the HALT\_REQ bit in the DGCS register
- encounters an error on either the PCI or VME buses.

Each of these conditions will cause the ACT bit to clear, and a corresponding status bit to be set in the DGCS register. If enabled in step 1, an interrupt will also be generated. Once the software has set the GO bit, the software can monitor for DMA completion by either waiting for generation of an interrupt, by polling the status bits in the DGCS register, or by polling the "processed" bits of the command packets. It is recommended that a background timer also be initiated to time-out the transfer. This will ensure that the DMA has not been hung up by a busy VMEbus, or other such system issues.

Linked-list operation can be halted by setting the HALT\_REQ bit in the DGCS register (Table A.38). When the HALT\_REQ bit is set, the DMA terminates when all transfers defined by the the current command packet is complete. It then loads the next command packet into its registers. The HALT bit in the DGCS register is asserted, and the ACT bit in the DGCS register is cleared. The DMA can be restarted by clearing the HALT status bit and setting the GO bit if desired during the same register write. If the DMA is indeed restarted, the ACT bit is set by the Universe and execution continues as if no HALT had occurred: i.e., the Universe processes the current command packet (see Figure 2.14 above).

In contrast to a halt, the DMA can also be immediately terminated through the STOP\_REQ bit. This will stop all DMA operations on the source bus immediately, and set the STOP bit in the same register when the last piece of queued data in the DMA FIFO has been written to the destination bus.

Once stopped, the DVA, DLA and DTBC registers contain values indicating the next addresses to read/write and the number of bytes remaining in the transfer. Clearing the STOP bit and setting the GO bit will cause the DMA to start up again from where it left off, including continuing with subsequent command packets in the list.

If the DMA is being stopped to insert a high priority DMA transfer, the remaining portion of the DMA transfer may be stored as a new command packet inserted at the top of the linked list. A new command packet with the attributes of the high priority transfer is then placed before that one in the list. Now the linked list is set up with the high priority packet first, followed by the remainder of the interrupted packet, followed in turn by the rest of the linked list. Finally, the DTBC register is cleared and the DCPD programmed with a pointer to the top of the list where the high priority command packet has been placed. When the GO bit is set (after clearing the STOP status bit in the DGCS register), the DMA will perform the transfers in the order set in the linked list. For more details on updating the linked list see “Linked List Updating” on page 2-80.

DMA transfers continue until the DMA encounters a command packet with the null bit set to “1”, indicating that the last packet has been reached. At this point, the DMA stops, the DONE bit is set, and the ACT flag is cleared. As it completes the transfers indicated by each command packet, the DMA sets the “processed” bit in that command packet before reading in the next command packet and processing its contents.

### 2.8.3.1 Linked List Updating

The Universe provides a mechanism which allows the linked list to be updated with additional linked list entries without halting or stopping the DMA. This takes place through the use of a semaphore in the device: the UPDATE bit in the D\_LLUE register (Table A.39). This bit ensures that a command packet is not altered for the linked-list update process at the same time that the DMA is reading the command packet into the DMA registers.

Adding to a linked list begins by writing a “1” to the UPDATE bit. The DMA checks this bit before proceeding to the next command packet. If the UPDATE bit is “0”, then the DMA locks the UPDATE bit against writes and proceeds to the next command packet. If the UPDATE bit is “1”, then the DMA waits until the bit is cleared before proceeding to the next command packet. Therefore, setting the UPDATE bit is a means of stalling the DMA at command packet boundaries while local logic updates the linked list.

In order to ensure that the DMA is not currently reading a command packet during updates, the update logic must write a “1” to the UPDATE bit and read a value back. If a “0” is read back from the UPDATE bit, then the DMA is currently reading a command packet and has locked the UPDATE bit against writes. If a “1” is read back from the UPDATE bit, then the DMA is idle or processing a transaction and command packets can be updated. If the DMA attempts to proceed to the next command packet during the update, it will encounter the set UPDATE bit and wait until the bit is cleared.

If a set of linked command packets has already been created with empty packets at the end of new transfers, adding to the end of the current linked list takes the following procedure:

1. Get UPDATE valid (write “1”, read back “1”),
2. Program attributes for new transfer in next available packet in list.
3. Change “null” pointer (on previous tail of linked list),
4. Release update (clear the UPDATE bit).

After updating the linked list, the DMA controller will be in three possible conditions:

1. It may be active and working its way through the linked list. In this case, no further steps are required.
2. The DMA may be idle (done) because it reached the final command packet. If a full set of linked command packets had already been created ahead of time, then the DCPD register will point to the most recently programmed command packet, and the DTBC register will be zero. The DMA can be started on the new packet by simply clearing the DONE bit and setting the GO bit in the DGCS register. If a set of command packets had not been created ahead of time, the DCPD register may not be programmed to any valid packet, and will need programming to the newly programmed packet.
3. The DMA has encountered an error. In this circumstance, see “DMA Error Handling” on page 2-86 for how to handle DMA errors.

Operation may be considerably simplified by ensuring that sufficient command packets have been created during system initialization, probably in a circular queue. In this fashion, when a new entry is added to the list, it is simply a matter of programming the next available entry in the list with the new transfer attributes and changing the previously last packet's "null" bit to zero. The DCPD register will be guaranteed to point to a valid command packet, so upon updating the list, both cases 1 and 2 above can be covered by clearing the DONE bit and setting the GO bit. This will have no effect for case 1 since the DMA is still active, and will restart the DMA for case 2.

If an error has been encountered by the DMA (case 3), setting the GO bit and clearing the DONE bit will not be sufficient to restart the DMA—the error bits in the DGCS register will also have to be cleared before operation can continue.

## 2.8.4 FIFO Operation and Bus Ownership

The DMA uses a 16-entry deep by 64-bit wide bi-directional FIFO to increase performance for DMA transfers. In general, the DMA reads data from the source, and stores it as transactions in the FIFO. On the destination side, the DMA requests ownership of the master and once granted begins transfers. Transfers stop on the source side when the FIFO fills, and on the destination side when the FIFO empties.

### 2.8.4.1 PCI to VME Transfers

On the PCI side, the PCI bus is requested for the current operation once the available space in the DMAFIFO is greater than or equal to the programmed PCI aligned burst size (PABS in the MAST\_CTL register). The DMA channel fills the DMAFIFO using PCI read transactions with each transaction broken at address boundaries determined by the programmed PCI aligned burst size. This ensures that the DMA makes optimal use of the PCI bus by always generating bursts of 32 or 64 bytes with zero wait states.

The DMA packs read data into the DMAFIFO to the full 64-bit width of the FIFO, independent of the width of the PCI bus, or the data width of the ensuing VMEbus transaction. The PCI read transactions continue until either the DMA has completed the full programmed transfer, or there is insufficient room available in the DMAFIFO for a full transaction. The available space required for another burst read transaction is eight entries if PABS is programmed to 64 bytes, or four entries if programmed to 32 bytes. Since the VMEbus is typically much slower than the PCI bus, the DMAFIFO may fill frequently during PCI to VME transfers. When the DMAFIFO fills, the PCI bus is free for other transactions (for example, between other devices on the bus or possibly for use by the Universe's VME Slave Channel). The DMA only resumes read transactions on the PCI bus when the DMAFIFO has space for another aligned burst size transaction.

Once eight entries have been queued into the FIFO, the DMA requests ownership of the Universe's VME master interface (see "VMEbus Requester" on page 2-6 on how the VME Master Interface is shared between the DMA, the PCI slave channel, and the Interrupt channel). The Universe maintains ownership of the master interface until:

- the DMAFIFO is empty,
- the DMA block is complete,
- the DMA is stopped,
- a linked list is halted,
- the DMA encounters an error, or
- the DMA VME tenure limit (VON in the DGCS register).



The DMA can be programmed to limit its VMEbus tenure to fixed block sizes using the VON field in the DGCS register (Table A.38). With VON enabled, the DMA will relinquish ownership of the master interface at defined address boundaries: 256 bytes, 512 bytes, 1 Kbyte, 2 Kbytes, 4 Kbytes, 8 Kbytes, or 16 Kbytes. By default, VON is disabled and the DMA VMEbus tenure will be limited by one of the other mechanisms listed in the previous paragraph.

To further control the DMA's VMEbus ownership, the VOFF timer in the DGCS register can be used to program the DMA to remain off the VMEbus for a specified period when VME tenure is relinquished. The VOFF timer can be set to 16  $\mu$ s, 32  $\mu$ s, 64  $\mu$ s, 128  $\mu$ s, 256  $\mu$ s, 512  $\mu$ s, or 1024  $\mu$ s and only becomes active when the DMA reaches a boundary determined by the VON counter. The timer is disabled by default. This timer can ensure a predetermined amount of VMEbus bandwidth is available to other VME devices (including the PCI slave channel and the Interrupt channel) during DMA transfers. The VOFF timer is not invoked when the DMAFIFO empties of by any other mechanism other than expiry of the VON byte count.

The DMA Channel unpacks the 64-bit data queued in the DMAFIFO to whatever the programmed transfer width is on the VMEbus (e.g. D16, D32, or D64). The VME Master Interface delivers the data in the DMAFIFO according to the VMEbus cycle type programmed into the DCTL register (Table A.33, see “DMA Controller” on page 2-67). The DMA provides data to the VMEbus until:

- the DMAFIFO empties, or
- the DMA VMEbus Tenure Byte Count (VON in the DMA\_GCSR register, Table A.38) expires.

If the DMAFIFO empties, transfers on the VMEbus stop and, if the cycle being generated is a block transfer, then the block is terminated (AS\* negated) and VMEbus ownership is relinquished by the DMA. The DMA does not re-request VMEbus ownership until another eight entries are queued in the DMAFIFO, or the DMA Channel has completed the current Transfer Block on the PCI bus (see “VMEbus Release” on page 2-8).

PCI bus transactions are the full width of the PCI data bus with appropriate byte lanes enabled. The maximum VMEbus data width is programmable to 8, 16, 32, or 64 bits. Byte transfers can be only of type DO8 (EO). Because the PCI bus has a more flexible byte lane enabling scheme than the VMEbus, the Universe may be required to generate a variety of VMEbus transaction types to handle the byte resolution of the starting and ending addresses (see “Data Transfer” on page 2-34).

### 2.8.4.2 VME to PCI Transfers

With DMA transfers in this direction, the DMA Channel begins to queue data in the DMAFIFO as soon as there is room for eight entries (64 bytes) in the DMAFIFO. When enough room is available, the DMA will request the VMEbus (through the VME master interface) and begin reading data from the VMEbus. The Universe maintains VMEbus ownership until:

- the DMAFIFO is full,
- the DMA block is complete,
- the DMA is stopped,
- a linked list is halted,
- the DMA encounters an error, or
- the VMEbus VME tenure limit is reached (VON in the DGCS register).

The DMA can be programmed to limit its VMEbus tenure to fixed block sizes using the VON field in the DGCS register (Table A.38). With VON enabled, the DMA will relinquish ownership of the master interface at defined address boundaries: 256 bytes, 512 bytes, 1 Kbyte, 2 Kbytes, 4 Kbytes, 8 Kbytes, or 16 Kbytes. By default, VON is disabled and the DMA VMEbus tenure will be limited by one of the other mechanisms listed in the previous paragraph.

To further control the DMA's VMEbus ownership, the VOFF timer in the DGCS register can be used to program the DMA to remain off the VMEbus for a specified period when VME tenure is relinquished. The VOFF timer can be set to 16  $\mu$ s, 32  $\mu$ s, 64  $\mu$ s, 128  $\mu$ s, 256  $\mu$ s, 512  $\mu$ s, or 1024  $\mu$ s and only becomes active when the DMA reaches a boundary determined by the VON counter. The timer is disabled by default. This timer can ensure a predetermined amount of VMEbus bandwidth is available to other VME devices (including the PCI slave channel and the Interrupt channel) during DMA transfers. The VOFF timer is not invoked when the DMAFIFO empties of by any other mechanism other than expiry of the VON byte count.

On the PCI side, entries in the DMAFIFO are delivered to the PCI bus as PCI write transactions as soon as the number of entries reaches the PCI aligned burst size (as programmed in the PABS bit in the MAST\_CTL register). If the PCI bus responds too slowly, the DMAFIFO runs the risk of filling before write transactions can begin at the PCI master interface. Once the DMAFIFO reaches a "nearly full" state (corresponding to three entries remaining) then the DMA requests that the VME master interface complete its pending operations and stop. The pending read operations typically fill the DMAFIFO. Once the pending VMEbus reads are completed (or the VON timer expires), the DMA relinquishes VMEbus ownership and only re-requests the VME master interface once eight entries again become available in the DMAFIFO. If the bus was released due to encountering a VON boundary, the bus is not re-requested until the VOFF timer expires.

PCI bus transactions are the full width of the PCI data bus with appropriate byte lanes enabled. The maximum VMEbus data width is programmable to 8, 16, 32, or 64 bits. Byte transfers can be only of type DO8 (EO). Because the PCI bus has a more flexible byte lane enabling scheme than the VMEbus, the Universe may be required to generate a variety of VMEbus transaction types to handle the byte resolution of the starting and ending addresses (see “Universe as PCI Slave” on page 2-33).

### 2.8.5 DMA Interrupts

The interrupt channel in the Universe handles a single interrupt sourced from the DMA channel which it routes to either the VME or PCI buses via the DMA bits in the LINT\_EN and VINT\_EN registers. There are six internal DMA sources of interrupts and these are all routed to this single interrupt. Each of these six sources may be individually enabled, and are listed in Table 2.16 below. Setting the enable bit enables the corresponding interrupt source.

**Table 2.16 : DMA Interrupt Sources and Enable Bits**

Interrupt Source	Enable Bit
Stop Request	INT_STOP
Halt Request	INT_HALT
DMA Completion	INT_DONE
PCI Bus Error	INT_LERR
VMEbus Error	INT_VERR
Protocol Error	INT_M_ERR

Once an enabled DMA interrupt has occurred, regardless of whether the LINT\_EN or VINT\_EN enable bits have been set, the corresponding DMA bit in the LINT\_STAT and VINT\_STAT registers are set. Each one must be cleared independently. Clearing one does not clear the other.

### 2.8.6 DMA Error Handling

While the DMA is operating normally, the ACT bit in the DGCS register will be set (Table A.38). Once the DMA has terminated, it will clear this bit, and set one of six status bits in the same register. The DONE bit will be set if the DMA completed all its programmed operations normally. If the user interrupted the DMA, either the STOP or HALT bits will be set. If an error has occurred, one of the remaining three bits, LERR, VERR, or P\_ERR, will be set. All six forms of DMA terminations can be optionally set to generate a DMA interrupt by setting the appropriate enable bit in the DGCS register (see “DMA Interrupts” on page 2-85).

LERR is set if the DMA encounters an error on the PCI bus: either a Master Abort or Target Abort. Bits in the PCI\_CSR register will indicate which of these conditions caused the error.

VERR is set if the DMA encounters a bus error on the VME bus. This will be exclusively through a detected assertion of BERR\* during a DMA cycle.

P\_ERR is asserted if the GO bit in the DGCS register is set to start the DMA, and the DMA has been improperly programmed either because the BM bit in the PCI\_CSR has not been set to enable PCI bus mastership, or the source and destination start addresses are not aligned (see “Source and Destination Addresses” on page 2-68).

If the bus error occurs on the source bus, all DMA operations terminate immediately on the source bus and any data previously queued within the DMAFIFO is written to the destination bus. Once the DMAFIFO empties, the error status bit is set and the DMA generates an interrupt (as enabled by INT\_LERR and INT\_VERR in the DGCS register—see “DMA Interrupts” on page 2-85).

If the error occurs on the destination bus, the DMA immediately halts any operations on that bus, but continues reading from the source bus until the DMAFIFO no longer has room for another transaction. After the DMA stops on the source bus (or all data relevant to the command packet is read), the error bit in the DGCS register is set and an interrupt asserted (if enabled).

The address at which the error occurred will be indicated by the addresses stored in the VME and PCI Address registers (DVA and DLA). After an error on the source bus, the source bus address register will contain the address at which the error occurred. This will be independent of the size or type of transfers being performed on the bus. After an error on the destination bus, the destination bus address register will contain the address at which the error occurred.

Independent of the error occurring on the destination or source bus, the DMA\_CTL register contains the attributes relevant to the particular DMA transaction. In addition, the DTBC register provides the number of bytes remaining to transfer on the PCI side.

After an error or DMA completion, the DMA can be restarted with GO. However, the DMA will re-start only after the various status bits are cleared (including the DONE bit). The status bits are cleared by writing 1 to them. This may be done before or while setting the GO bit.

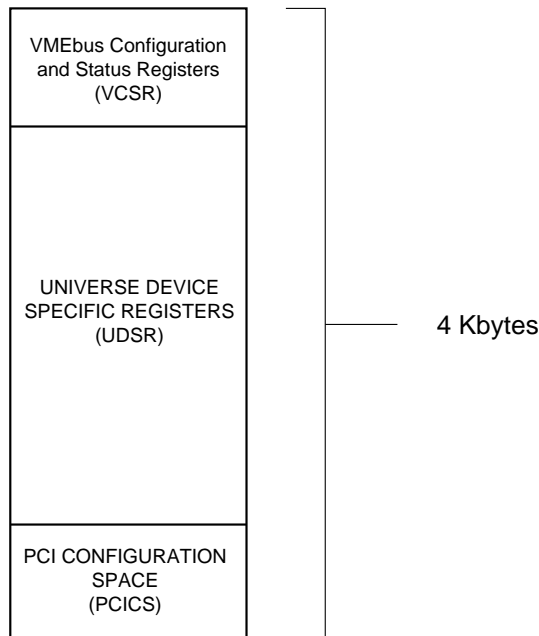
## 2.9 Registers

The Universe Control and Status Registers (UCSR) occupy 4 Kbytes of internal memory. This 4 Kbytes is logically divided into three groups (see Figure 2.16 below):

- PCI Configuration Space (PCICS),
- Universe Device Specific Registers (UDSR), and
- VMEbus Control and Status Registers (VCSR).

The Universe registers are little-endian.

The access mechanisms for the UCSR are different depending upon whether the register space is accessed from the PCI bus or VMEbus. Register access from the PCI bus and VMEbus is discussed below.



**Figure 2.16 : Universe Control and Status Register Space**

## 2.9.1 Register Access from the PCI Bus

There are two mechanisms to access the UCSR space from the PCI bus: through Configuration space or through the PCI defined base address register (Table A.6).

### 2.9.1.1 PCI Configuration Access

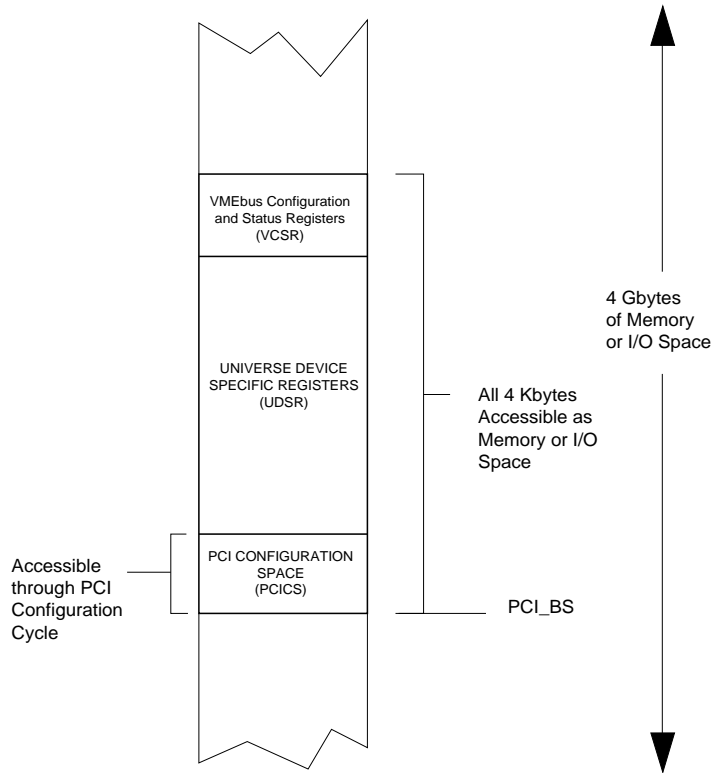
When the UCSR space is accessed as Configuration space, it means that the access is externally decoded and the Universe is notified via IDSEL# (much like a standard chip select signal). Since the register location is encoded by a 6-bit register number (a value used to index a 32-bit chunk of configuration space), only the lower 256 bytes of the UCSR can be accessed as configuration space (this corresponds to the PCICS in the UCSR space, see Figure 2.17 on page 2-89). Thus, only the PCI configuration registers are accessible through PCI Configuration cycles.

To access the other UCSR registers through Memory or I/O space, the base address of the PCI\_BS register (Table A.6) must first be programmed, since in Memory or I/O space the UCSRs are located as address offsets from the base address programmed into the BS field in the PCI\_BS register.

### 2.9.1.2 Memory or I/O Access

The UCSR space can be configured as Memory or I/O space at power-up, but not both (see “Power-Up Options” on page 2-98). There is an internal inversion on the power-up pin (VA[1]). When the VA[1] pin is sampled low at power-up, the device starts-up in I/O space and the SPACE bit of the PCI\_BS register (Table A.6) is set to 1. When VA[1] is sampled high at power-up, the Universe starts-up in Memory space. The user is free to locate the registers anywhere in the 32-bit Memory or I/O space; however, the registers are not prefetchable. Once configured at power-up, the UCSR space can not be reconfigured until the next power-up sequence, although their location within that space may be reprogrammed at any time.

All of the UCSR space (4 Kbytes) is accessible as Memory or I/O space from the PCI bus (see Figure 2.17 below) once the PCI\_BS register (Table A.6) has been programmed. The Base Address field of the PCI\_BS register can be programmed through PCI Configuration space (as described above) or through a VME access.



**Figure 2.17 : PCI Bus Access to UCSR as Memory or I/O Space**

### 2.9.2 Register Access from the VMEbus

There are two mechanisms to access the UCSR space from the VMEbus. One method uses a VMEbus Register Access Image (VRAI) which allows the user to put the UCSR in an A16, A24 or A32 address space. The VRAI approach is useful in systems not implementing CR/CSR space as defined in the VME64 specification. The other way to access the UCSR is as CR/CSR space, where each slot in the VMEbus system is assigned 512 Kbytes of CR/CSR space.

Each method is discussed below.

### 2.9.2.1 VMEbus Register Access Image

The VMEbus register access image is defined by the following register fields:

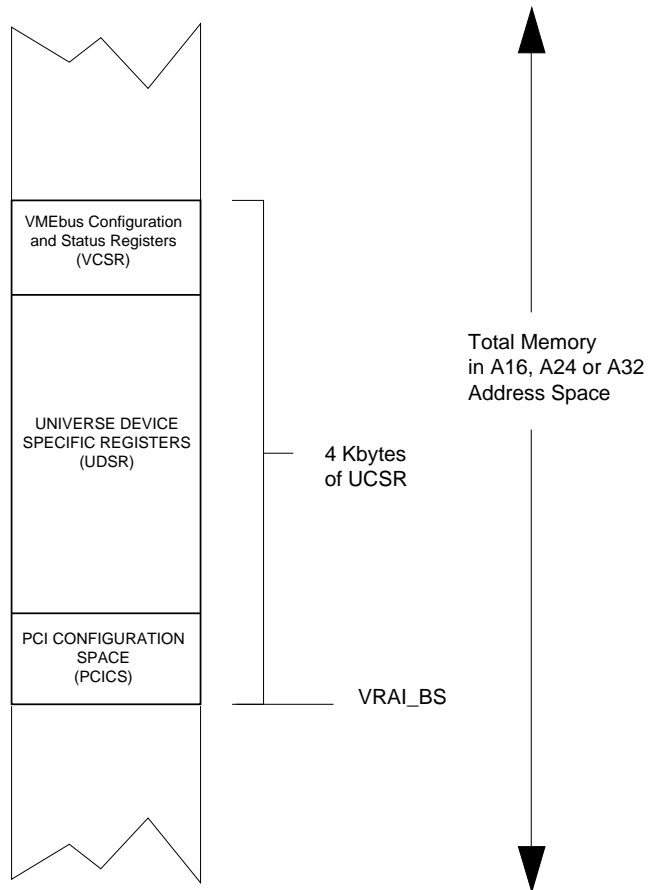
**Table 2.17 : Programming the VMEbus Register Access Image**

Field	Register Bits	Description
address space	VAS in Table A.76	one of A16, A24, A32
base address	BS[31:12] in Table A.77	lowest address in the 4Kbyte slave image
slave image enable	EN in Table A.76	enables VMEbus register access image
mode	SUPER in Table A.76	Supervisor and/or Non-Privileged
type	PGM in Table A.76	Program and/or Data

The VMEbus Register Access Image occupies 4 Kbytes in A16, A24 or A32 space (depending upon the programming of the address space described in Table 2.17 above, see Figure 2.18 below). All registers are accessed as address offsets from the VRAI base address programmed in the VRAI\_BS register (Table A.77). The image can be enabled or disabled using the EN bit in the VRAI\_CTL register (Table A.76).

Note that the VRAI base address can be configured as a power-up option (see “Power-up Option Descriptions” on page 2-99).



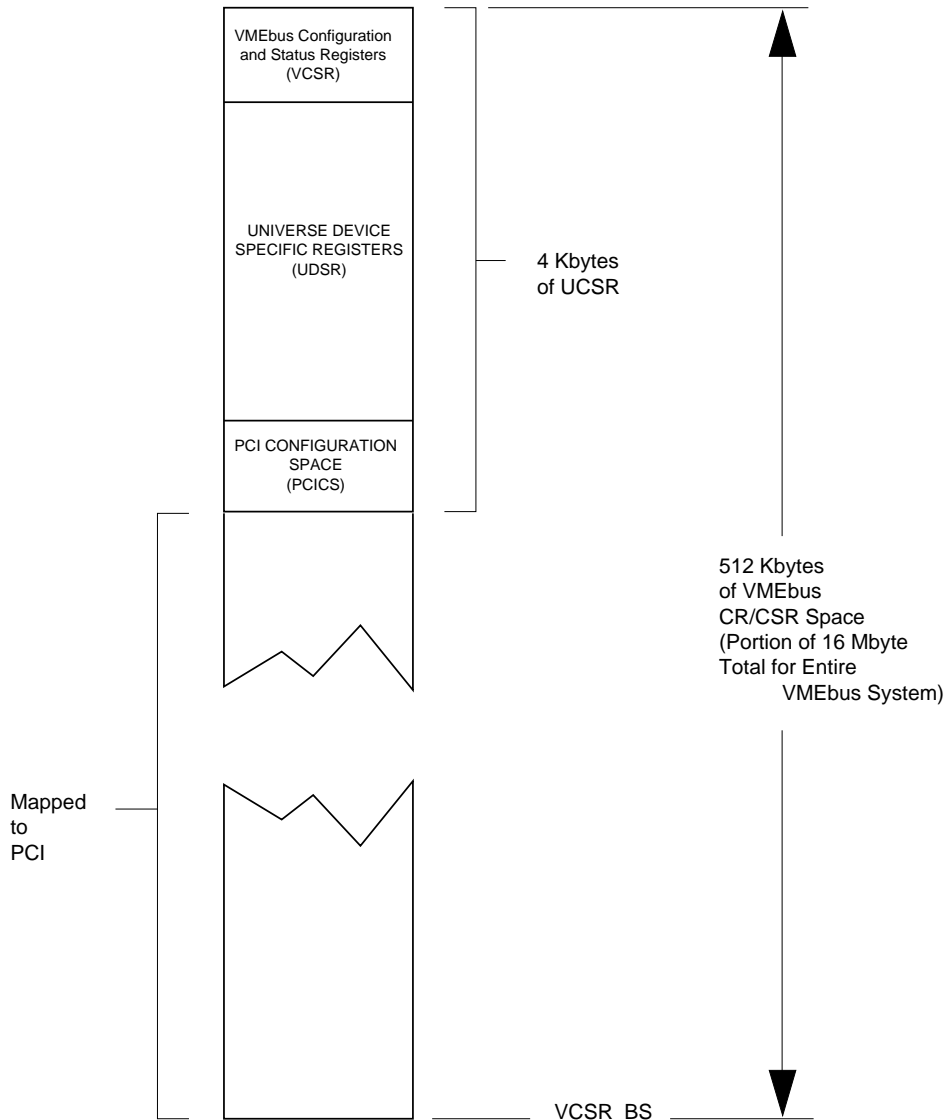


**Figure 2.18 : UCSR Access from the VMEbus Register Access Image**

### 2.9.2.2 CR/CSR Accesses

The VME64 specification assigns a total of 16 Mbytes of CR/CSR space for the entire VME system. The CR/CSR image is enabled with the EN bit in the VCSR\_CTL register (Table A.78). This 16 Mbytes is broken up into 512 Kbytes per slot for a total of 32 slots. The first 512 Kbyte block is reserved for use by the Auto-ID mechanism. The UCSR space occupies the upper 4 Kbytes of the 512 Kbytes available for its slot position (see Figure 2.19 below). The base address of the CR/CSR space allocated to the Universe's slot is programmed in the VCSR\_BS register (Table A.84). For CSRs not supported in the Universe and for CR accesses, the LAS field in the VCSR\_CTL register specifies the PCI bus command that is generated when the cycle is mapped to the PCI bus. There is also a translation offset added to the 24-bit VMEbus address to produce a 32-bit PCI bus address (programmed in the VCSR\_TO register, Table A.79).

Note that the registers in the UCSR space are located as address offsets from VCSR\_BS. These offsets are different from those used in the VRAI mechanisms, where the first register in the UCSR has address offset of zero (see Table A.1 in Appendix A). When accessing the UCSR in CR/CSR space, the first register will have an address offset of 508 Kbytes (512 Kbytes minus 4 Kbytes). A simple approach for determining the register offset when accessing the UCSR in CR/CSR space is to add 508 Kbytes (0x7F000) to the address offsets given in Table A.1.



**Figure 2.19 : UCSR Access in VMEbus CR/CSR Space**

## 2.10 Utility Functions

### 2.10.1 Resets

This section is divided in three sections. The first section lists the pins and registers that are involved in the reset circuitry. The second section presents and explains the reset circuitry diagram. The third section provides important suggestions and warnings about configuring the Universe reset circuitry.

#### 2.10.1.1 Overview of Reset Support

The Universe provides a number of pins and registers for reset support. Pin support is summarized in Table 2.18.

**Table 2.18 : Universe Signals Involved in Reset Circuitry**

Interface and direction	Pin name	Long Name	Brief Description <sup>a</sup>
JTAG Input	TRST#	JTAG Test Reset	Provides asynchronous initialization of the TAP controller in the Universe.
VME Input	VRSYSRST#	VMEbus Reset Input	Asserts LRST# on the local bus, resets the Universe, and reconfigures power-up options.
VME Output	VXSYSRST	VMEbus System Reset	Universe output for SYSRST* (resets the VMEbus)
PCI Input	PWRST#	Power-up Reset	Resets the Universe and reconfigures power-up options.
	RST#	PCI Reset Input	Resets the Universe from the PCI bus.
	VME_RESET# <sup>b</sup>	VME Reset Initiator	Causes Universe to assert VXSYSRST
PCI Output	LRST#	PCI Bus Reset Output	Resets PCI resources

*a. A more detailed account of the effects of reset signals is provided in “Reset Implementation Issues” on page 2-97*

*b. VME\_RESET# is an optional pin.*

While the Universe provides register support for resets (Table 2.19), the Universe is only reset by reset input pins. In order to reset the Universe by writing to registers, the Universe reset outputs must be connected to the Universe reset inputs. For example, the SW\_LRST bit in the MISC\_CTL register, which asserts the LRST# output, will not reset the Universe itself unless LRST# causes an assertion of RST#. As described in “Reset Implementation Issues” on page 2-97, caution must be exercised in order not to connect the Universe in a loop such that the Universe may be driven in a permanent reset.

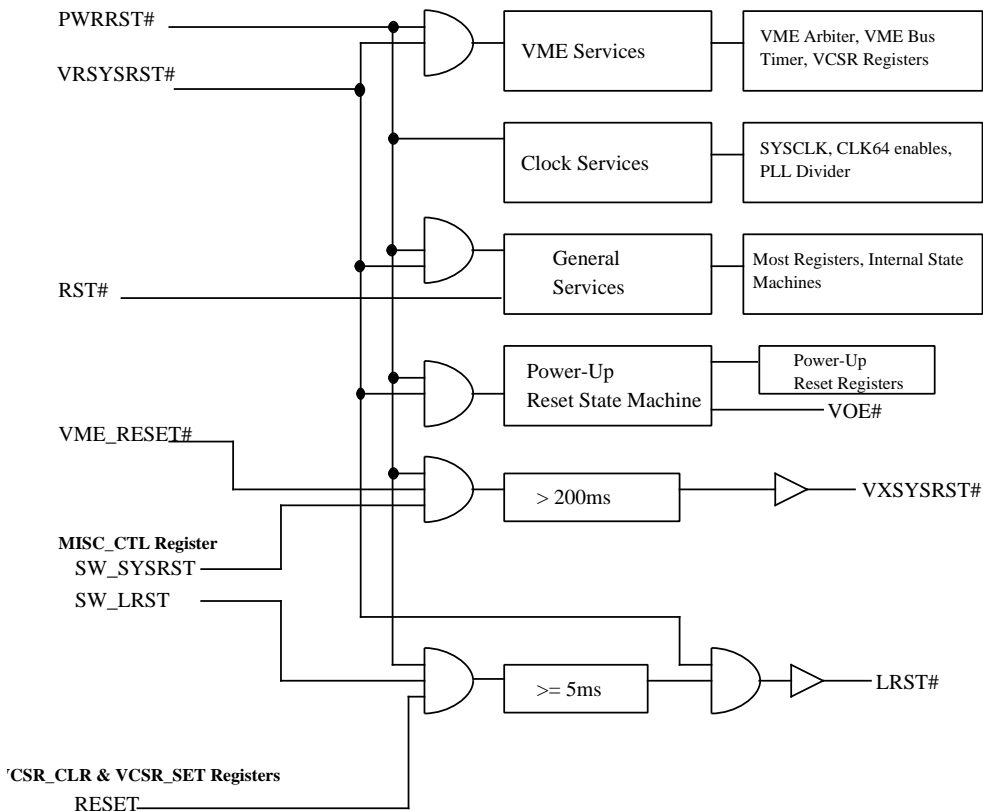
**Table 2.19 : Universe Register Support for Resets**

Register and Table	Name	Type	Function
MISC_CTL Table A.57	SW_LRST	W	Software PCI Reset 0=No affect, 1=Initiate LRST# A read always returns 0.
	SW_SYSRST	W	Software VMEbus SYSRESET 0=No affect, 1=Initiate SYSRST* A read always returns 0.
VCSR_SET Table A.83	RESET	R/W	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=assert LRST#
	SYSFAIL	R/W	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=assert VXSYSFAIL
VCSR_CLR Table A.82	RESET	R/W	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=negate LRST#
	SYSFAIL	R/W	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes:0=no effect, 1=negate VXSYSFAIL

More detailed information about the effects of various reset events is provided in the next section.

## 2.10.2 Universe Reset Circuitry

Figure 2.20 below details the categories of effects of each reset event. For example, it shows that when the PWRRST# signal is asserted, the VME Services, Clock Services, General Services, UCSR and power-up options are all reset, while VXYSRST# and LRST# are asserted. Figure 2.20 lists the members of these categories of reset effects. For example, the VME Services category is composed of the VME Arbiter, VME Bus Timer, and VCSR registers. Thus, when PWRRST# or VRSYSRST# is asserted, all of the said VME Services are reset.



**Figure 2.20 : Functional Block Diagram of Reset Circuitry**

Figure 2.20 above also indicates the reset effects that are extended in time. For example, VXYSRST# remains asserted for 256 ms after all initiators are removed - this satisfies

VMEbus rule 5.2 (minimum of 200 ms SYSRST\*). The external 64 MHz clock controls this assertion time.. LRST# is asserted for 5ms or more from all sources except VRSYSRST#.

Figure 2.20 also illustrates the effects of register events. For example, it shows how asserting the SW\_LRST bit in the MISC\_CTL register causes LRST# to be asserted for at least 5ms.

### 2.10.3 Reset Implementation Issues

The designer may wish to disallow the Universe from resetting the PCI bus, in which case the LRST# output may be left unconnected. Otherwise, LRST# should be grouped with other PCI reset generators to assert the RST# signal such that:

$RST\# = LRST\# \& \text{reset\_source1} \& \text{reset\_source2} \& \dots$

If the Universe is the only initiator of PCI reset, LRST# may be directly connected to RST#.

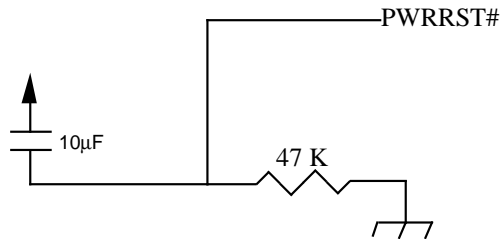
Assertion of VME\_RESET# causes the Universe to assert VXSYSRST.



*Caution: Since VME\_RESET# causes assertion of SYSRST\*, and since SYSRST\* causes assertion of LRST#, tying both VME\_RESET# and LRST# to RST# will put the Universe into permanent reset. If VME\_RESET# is to be driven by PCI reset logic, ensure that the logic is designed to break this feedback path.*

The Universe supports the VMEbus CSR bit clear and bit set registers (Table A.82 and Table A.83). The VCSR\_SET registers allows the user to set LRST# or SYSFAIL by writing to the RESET or SYSFAIL bits, respectively. The PCI reset or SYSFAIL assertion remains in effect until the corresponding bit is written to in the VCSR\_CLR register. The FAIL bit in each of these registers is a status bit and is set by the software to indicate board failure.

The PWRRST# input is used to provide reset to the Universe until the power supply has reached a stable level. It should be held asserted for over 100 milliseconds after power is stable. Typically this can be achieved through a resistor/capacitor combination (see Figure 2.21); however, a more accurate solution using under voltage sensing circuits (i.e. MC34064) is often implemented.



**Figure 2.21 : PWRRST# assertion**

#### 2.10.4 Power-Up Options

The majority of the Universe power-up options (listed below) are loaded from the VMEbus address and data lines after any PWRRST# or SYSRST\* event. Two power-up options are not initiated by PWRRST#. The first of these is PCI bus width (a power-up option required by the PCI bus specification), and this is loaded on any RST# event from the REQ64# pin. The second special power-up option is VMEbus SYSCON enabling, required by the VMEbus specification. The SYSCON option is loaded during a SYSRST\* event from the BG3IN\* signal.

All power-up options are loaded during their particular reset event from the state of a particular pin or group of pins. Each of these pins has internal pull-downs to put the Universe into a default configuration. If other than the default is required, a pull-up of approximately 10kΩ (or active drive) is required on the signal. These pull-ups are only necessary on the general group of power-up options since the PCI and VMEbus specifications provide for the appropriate levels to be actively driven on the REQ64# and BG3IN\* signals, respectively.



The pull-ups for the general power-up options (if other than default values are required) must be placed on the VA[31:1], VLWORD, and VD[31:27] lines. During reset, the Universe will negate VOE#, putting these signals into a high-impedance state. The pull-ups (or internal pull-downs) will bring the option pins to their appropriate state. Within two CLK64 periods after both SYSRST\* and PWRRST# are negated, the Universe latches the levels on the option pins, and then negates VOE# one clock later. This enables the VMEbus transceivers inwards.

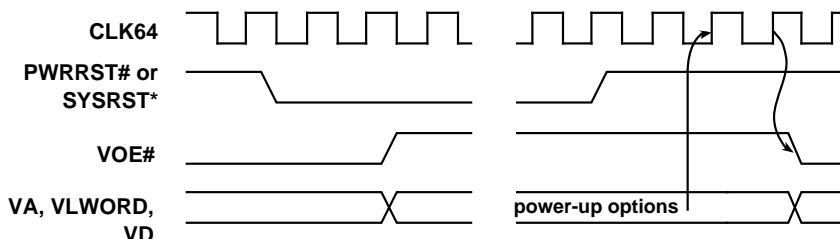


Figure 2.22 : Power-up Options Timing



*Note that because of the operation of the power-up options, the VME buffers are not enabled until three CLK64 periods after release of SYSRST\* (approximately 45ns). Including worst case backplane skew of 25ns implies that the Universe will not be prepared to receive a slave access until 70ns after release of SYSRST\*.*

### 2.10.4.1 Power-up Option Descriptions

#### VME Register Access Image

The Universe is provided with multiple VMEbus slave images, each of which may provide a different mapping of VMEbus cycles to PCI cycles. All VMEbus slave images are configurable through a set of VME slave registers: VSIxCTL, VSIx\_BS, VSIx\_BA, and VSIx\_IO. Typically these registers are configured by software, but until this is completed no VMEbus to PCI accesses are possible.

The VME register access image power-up option permits access from the VMEbus to the Universe internal registers at power-up. The power-up option allows programming of the VME register slave image address space and the upper five bits of its base address; all other bits will be zero (see Table 2.20 below). Once access is provided to the registers, then all other Universe features (such as further VMEbus slave images) can be configured from the VMEbus.

**Table 2.20 : VRAI Base Address Power-up Options**

VRAI_CTL: VAS	BS [31:24]	BS [23:16]	BS [15:12]
A16	0	0	Power-up Option VA [28:25]
A24	0	Power-up Option VA [28:21]	0
A32	Power-up Option VA [28:21]	0	0

Table 2.20 above shows how the upper bits in the VRAI base address are programmed for A16, A24, and A32 VMEbus register access images.

### VME CR/CSR Slave Image

CR/CSR space is a new address space introduced with the VME64 specification. The CR/CSR space on any VME device is 512 Kbytes in size: the upper region of the 512 Kbytes dedicated to register space, and the lower region is dedicated to configuration ROM. The Universe maps its internal registers to the upper region of the CR/CSR space, and passes all other accesses through to the PCI bus (see “Registers” on page 2-87).

The VME CR/CSR Slave Image power-up option maps CR/CSR accesses to the PCI bus. CR/CSR space can be mapped to memory or I/O space with a 5-bit offset. This allows mapping to any 128Mbyte page on the PCI bus. As part of this implementation, ensure that the Universe's PCI master interface is enabled through the MAST\_EN bit power-up option (see below) or configured through a register access before accessing configuration ROM.

### PCI Slave Image

The PCI Slave Image power-up option provides for default enabling of a PCI slave image (automatically mapping PCI cycles to the VMEbus). The default slave image can be mapped with base and bounds at 256MB resolution in Memory or I/O space, and map PCI transactions to different VME address spaces. Beyond the settings provided for in this power-up option, the slave image will possess its other default conditions: the translation offset will be zero, posted writes will be disabled, and only 32-bit (maximum) non-block VME cycles in the non-privileged data space will be generated. This option would typically be used to access permits the use of Boot ROM on another card in the VMEbus system.

### PCI CSR Image Space

This power-up option maps the Universe's internal PCI registers to either Memory or I/O space on the PCI bus. Once set, this mapping persists until the next power-up sequence.

### PCI Bus Width

The Universe's PCI interface can be used as a 32-bit bus or 64-bit bus. If used as a 32-bit interface, the 64-bit pins, AD[32:63] and ACK64# are left unterminated (REQ64# must be pulled-up at reset for the Universe to be configured with a 32-bit PCI bus, see "32-Bit Versus 64-Bit PCI" on page 2-25). This power-up option, required by the PCI specification, provides the necessary information to the Universe so that these unused pins may be left unterminated.

### Auto-ID

There are two Auto-ID mechanisms provided by the Universe. One is the VME64 specified version which relies upon use of the CR/CSR space for configuration of the VME system, and a Newbridge proprietary system which uses the IACK daisy chain for identifying cards in a system. Either of these mechanisms can be enabled at power-up (see "Automatic Slot Identification" on page 2-20).

Because VME64 Auto-ID relies upon SYSFAIL to operate correctly, this power-up option overrides the SYSFAIL power-up option described below.

### BI-Mode

BI-Mode (Bus Isolation Mode) is a mechanism for logically isolating the Universe from the VMEbus for diagnostic and maintenance purposes. BI-mode may be enabled as a power-up option (see "BI-Mode®" on page 2-24).

### SYSFAIL\* Assertion

This power-up option causes the Universe to assert SYSFAIL\* immediately upon entry into reset. The SYSFAIL\* pin is released through a register access. Note that this power-up option is over-ridden if VME64 Auto-ID has been enabled. This option would be used when extensive on-board diagnostics need to be performed before release of SYSFAIL\*. After completion of diagnostics, SYSFAIL\* may be released through software or through initiation of the VME64 Auto-ID sequence if that mechanism is to be used (see "Auto Slot ID: VME64 Specified" on page 2-20).

**Table 2.21 : Power-Up Options**

Option	Enabled by	Register	Field	Default	Pins
VME Register Access Slave Image	PWR, SYS	VRAI_CTL	EN	disabled	VA[31]
			VAS	A16	VA[30:29]
		VRAI_BS	BS	0x00	VA[28:21]
VME CR/CSR slave image	PWR, SYS	VCSR_CTL	LAS[0] <sup>a</sup>	memory	VA[20]
		VCSR_TO	TO	0x00	VA[19:15]
		PCI_CSR	BM	disabled	VA[14]
PCI Slave Image	PWR, SYS	LSIO_CTL	EN	disabled	VA[13]
			LAS[0]	memory	VA[12] †
			VAS	A16	VA[11:10]
		LSIO_BS	BS	0x0	VA[9:6]
		LSIO_BD	BD	0x0	VA[5:2]
PCI Register Access	PWR, SYS	PCI_BS	SPACE	I/O	VA[1]
PCI Bus Size	RST#	MISC_STAT	LCLSIZE		REQ64#
Package Option	PWR, SYS	MISC_STAT	PKGSIZE	64-bit package	VD[31]
Auto-ID	PWR, SYS	MISC_STAT	DY4AUTO	disabled	VD[30]
		MISC_CTL	V64AUTO	disabled	VD[29]
BI-Mode®	PWR, SYS	MISC_CTL	BI	disabled	VD[28]
SYSFAIL* Assertion	PWR, SYS	VCSR_SET	SYSFAIL	asserted	VD[27]
		VCSR_CLR	SYSFAIL		
Auto-Syscon Detect	SYS	MISC_CTL	SYSCON	enabled	BGIN[3]*

a. The LAS field will enable the PCI\_CSR register's MS or IOS field if the EN FIELD of the LSIO\_CTL register is set.

### 2.10.5 Hardware Initialization (Normal Operating Mode)

For Universe normal system operation and to set the JTAG controller to a neutral state, the following Universe manufacturing test pins should be tied in accordance with Table 2.22 below.

**Table 2.22 : Manufacturing Pin Requirements for Normal Operating Mode**

Pin Name	Pin Value
TMODE[2]	V <sub>SS</sub> (or pulled-down if board tests will occasionally be performed, see “Auxiliary Test Modes” on page 2-104)
TMODE[1]	
TMODE[0]	
PLL_TESTSEL	V <sub>SS</sub>
ENID	V <sub>SS</sub>
PLL_TESTOUT	N/C
VOCCTL	V <sub>SS</sub>

## 2.10.6 Support for board testing

The Universe provides two levels of support for board testing, auxiliary modes (NAND tree simulation and High Impedance) and JTAG. These two topics are discussed in turn.

### 2.10.6.1 Auxiliary Test Modes

Two auxiliary test modes are supported: NAND tree simulation and high impedance. The Universe has three test mode input pins (TMODE[2:0]). For normal operations these inputs should be tied to ground (or pulled to ground through resistors). Table 2.23 below indicates the 3 operating modes of the Universe. At reset the TMODE[2:0] inputs are latched by the Universe to determine the mode of operation. The Universe remains in this mode until the TMODE[2:0] inputs have changed and a reset event has occurred. PLL\_TESTSEL must be high for any test mode.

**Table 2.23 : Test Mode Operation**

Operation Mode	TMODE[2:0]	PLL_TESTSEL
Normal Mode	000	0
Reserved	001	0
Reserved	010	1
Reserved	011	1
NAND Tree Simulation	100	1
Reserved	101	1
High Impedance	110	0/1
Reserved	111	1

For NAND Tree Simulation, the values of the TMODE pins are latched during the active part of  $\overline{\text{PWRRST}}$ . These pins can change state during the Nand Tree tests. The timers are always accelerated in this mode. All outputs are tristated in this mode, except for the VXSYSFAIL output pin.

For High Impedance mode, the values of the TMODE pins are also latched during the active part of  $\overline{\text{PWRRST}}$ . All outputs are tristated in this mode, except for the VXSYSFAIL output pin.

### 2.10.6.2 JTAG support

The Universe includes dedicated user-accessible test logic that is fully compatible with the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. This standard was developed by the Test Technology Technical Committee of IEEE Computer Society and the Joint Test Action Group (JTAG). The Universe's JTAG support includes:

- A five-pin JTAG interface (TCK, TDI, TDO, TMS, and TRST#),
- A JTAG TAP controller,
- A three-bit instruction register,
- A boundary scan register,
- A bypass register,
- And an ICODE register.

The following required public instructions are supported: BYPASS (3'b111), SAMPLE(3'b100), and EXTEST(3'b000). The optional public instruction IDCODE(3'b011) selects the IDCODE register which returns 32'b01e201d. The following external pins are not part of the boundary scan register: LCLK, PLL\_TESTOUT, PLL\_TESTSEL, TMODE[3:0], AND VCOCTL.

A BSDL file is available upon request from Tundra Semiconductor Corporation.

### 2.10.7 CLK64

CLK64 is a 64 MHz clock that is required by the Universe in order to synchronize internal Universe state machines and to produce the VMEbus system clock (VSYSCLK) when the Universe is system controller (SYSCON). This clock is specified to have a minimum 60-40 duty cycle with a maximum rise time of 5ns.





## 3 Description of Signals

The following detailed description of the Universe signals is organized according to these functional groups:

- VMEbus Signals
- PCI Signals

### 3.1 VMEbus Signals

CLK64	Input
Reference Clock – this 64MHz clock is used to generate fixed timing parameters. It requires a 50-50 duty cycle ( $\pm 20\%$ ) with a 5ns maximum rise time. CLK64 is required to synchronize the internal state machines of the VME side of the Universe.	
VA [31:1]	Bidirectional
VMEbus Address Lines 31 to 01 – during MBLT transfers, VA 31-01 serve as data bits D63-D33. VA03-01 are used to indicate interrupt level on the VMEbus.	
VA_DIR	Output
VMEbus Address Transceiver Direction Control – the Universe controls the direction of the address (VA31-01, VLWORD#) transceivers as required for master, slave and bus isolation modes. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
VAM [5:0]	Bidirectional
VMEbus Address Modifier Codes – these codes indicate the address space being accessed (A16, A24, A32), the privilege level (user, supervisor), the cycle type (standard, BLT, MBLT) and the data type (program, data).	
VAM_DIR	Output
VMEbus AM Code Direction Control – controls the direction of the AM code transceivers as required for master, slave and bus isolation modes. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
VAS#	Bidirectional
VMEbus Address Strobe – the falling edge of VAS# indicates a valid address on the bus. By continuing to assert VAS#, ownership of the bus is maintained during a RMW cycle.	

### 3.1 VMEbus Signals (Continued)

VAS_DIR	Output
VMEbus Address Strobe Direction Control – controls the direction of the address strobe transceiver as required for master, slave and bus isolation modes. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
VBCLR#	Output
VMEbus Bus Clear – requests that the current owner release the bus. Asserted by the Universe when configured as SYSCON and the arbiter detects a higher level pending request.	
VBGI# [3:0]	Input
VMEbus Bus Grant Inputs – The VME arbiter awards use of the data transfer bus by driving these bus grant lines low. The signal propagates down the bus grant daisy chain and is either: <ul style="list-style-type: none"> <li>accepted by a requester if it requesting at the appropriate level, or</li> <li>passed on as a VBGO [3:0]# to the next board in the bus grant daisy chain.</li> </ul>	
VBGO# [3:0]	Output
VMEbus Bus Grant Outputs – Only one output is asserted at any time, according to the level at which the VMEbus is being granted.	
VD [31:0]	Bidirectional
VMEbus Data Lines 31 through 00	
VD_DIR	Output
VMEbus Data Transceiver Direction Control – the Universe controls the direction of the data (VD [31:0]) transceivers as required for master, slave and bus isolation modes. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
VDS# [1:0]	Bidirectional
VMEbus Data Strokes – the level of these signals are used to indicate active byte lanes: <ul style="list-style-type: none"> <li>During write cycles, the falling edge indicates valid data on the bus.</li> <li>During read cycles, assertion indicates a request to a slave to provide data.</li> </ul>	
VDS_DIR	Output
VMEbus Data Strobe Direction Control – controls the direction of the data strobe transceivers as required for master, slave and bus isolation modes. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
VDTACK#	Bidirectional
VMEbus Data Transfer Acknowledge – VDTACK# driven low indicates that the addressed slave has responded to the transfer.	

### 3.1 VMEbus Signals (Continued)

<b>VIACK#</b>	<b>Bidirectional</b>
VMEbus Interrupt Acknowledge – indicates that the cycle just beginning is an interrupt acknowledge cycle.	
<b>VIACKI#</b>	<b>Input</b>
VMEbus Interrupt Acknowledge In – Input for IACK daisy chain driver. If interrupt acknowledge is at same level as interrupt currently generated by the Universe, then the cycle is accepted. If interrupt acknowledge is not at same level as current interrupt or Universe is not generating an interrupt, then the Universe propagates VIACKO#.	
<b>VIACKO#</b>	<b>Output</b>
VMEbus Interrupt Acknowledge Out– generated by the Universe if it receives VIACKI# and is not currently generating an interrupt at the level being acknowledged.	
<b>VLWORD#</b>	<b>Bidirectional</b>
VMEbus Longword Data Transfer Size Indicator – this signal is used in conjunction with the two data strobes VDS [1:0]# and VA 01 to indicate the number of bytes (1 – 4) in the current transfer. During MBLT transfers VLWORD# serves as data bit D32.	
<b>VOE#</b>	<b>Output</b>
VMEbus Transceiver Output Enable – used to control transceivers to isolate the Universe from the VMEbus during a reset or BI-mode. On power-up, VOE# is high (to disable the buffers).	
<b>VRACFAIL#</b>	<b>Input</b>
VMEbus ACFAIL Input signal – warns the VMEbus system of imminent power failure. This gives the modules in the system time to shut down in an orderly fashion before powerdown. ACFAIL is mapped to a PCI interrupt.	
<b>VRBBSY#</b>	<b>Input</b>
VMEbus Receive Bus Busy – allows the Universe to monitor whether the VMEbus is owned by another VMEbus master	
<b>VRBERR#</b>	<b>Input</b>
VMEbus Receive Bus Error – a low level signal indicates that the addressed slave has not responded, or is signalling an error.	
<b>VRBR# [3:0]</b>	<b>Input</b>
VMEbus Receive Bus Request Lines – if the Universe is the Syscon, the Arbiter logic monitors these signals and generates the appropriate Bus Grant signals. Also monitored by requester in ROR mode.	
<b>VRIRQ# [7:1]</b>	<b>Input</b>
VMEbus Receive Interrupts 7 through 1 – these interrupts can be mapped to any of the Universe's PCI interrupt outputs. VRIRQ7-1# are individually maskable, but cannot be read.	
<b>VRSYSFAIL#</b>	<b>Input</b>
VMEbus Receive SYSFAIL – asserted by a VMEbus system to indicate some system failure. VRSYSFAIL# is mapped to a PCI interrupt.	

### 3.1 VMEbus Signals (Continued)

<b>VRSYSRST#</b>	<b>Input</b>
VMEbus Receive System Reset – causes assertion of LRST# on the local bus and resets the Universe.	
<b>VSLAVE_DIR</b>	<b>Output</b>
VMEbus Slave Direction Control – transceiver control that allow the Universe to drive DTACK* on the VMEbus. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
<b>VSYSCLK</b>	<b>Bidirectional</b>
VMEbus System Clock – generated by the Universe when it is the Syscon and monitored during DY4 Auto ID sequence	
<b>VSCON_DIR</b>	<b>Output</b>
Syscon Direction Control – transceiver control that allows the Universe to drive VBCLR# and SYSCLK. When the Universe is driving lines on the VMEbus, this signal is driven high; when the VMEbus is driving the Universe, this signal is driven low.	
<b>VWRITE#</b>	<b>Bidirectional</b>
VMEbus Write signal – indicates the direction of data transfer.	
<b>VXBBSY</b>	<b>Output</b>
VMEbus Transmit Bus Busy Signal – generated by the Universe when it is VMEbus master	
<b>VXBERR</b>	<b>Output</b>
VMEbus Transmit Bus Error Signal – generated by the Universe when PCI target generates target abort on coupled PCI access from VMEbus.	
<b>VXBR [3:0]</b>	<b>Output</b>
VMEbus Transmit Bus Request – the Universe requests the VMEbus when it needs to become VMEbus master.	
<b>VXIRQ [7:1]</b>	<b>Output</b>
VMEbus Transmit Interrupts – the VMEbus interrupt outputs are individually maskable.	
<b>VXSYSFAIL</b>	<b>Output</b>
VMEbus System Failure – asserted by the Universe during reset and plays a role in VME64 Auto ID	
<b>VXSYSRST</b>	<b>Output</b>
VMEbus System Reset – the Universe output for SYSRST*.	

## 3.2 PCI Bus Signals

<b>ACK64#</b>	<b>Bidirectional</b>
Acknowledge 64-bit Transfer – when driven by the PCI slave (target), it indicates slave can perform a 64-bit transfer.	
<b>AD [31:0]</b>	<b>Bidirectional</b>
PCI Address/Data Bus – address and data are multiplexed over these pins providing a 32-bit address/data bus.	
<b>AD [63:32]</b>	<b>Bidirectional</b>
PCI Address/Data Bus – address and data are multiplexed over these pins providing 64-bit address and data capability.	
<b>C/BE# [7:0]</b>	<b>Bidirectional</b>
PCI Bus Command and Byte Enable Lines – command and byte enable information is multiplexed over all eight C/BE lines. C/BE [7:4]# are only used in a 64-bit PCI bus	
<b>DEVSEL#</b>	<b>Bidirectional</b>
PCI Device Select – is driven by the Universe when it is accessed as PCI slave.	
<b>ENID</b>	<b>Input</b>
Enable IDD Tests – required for ASIC manufacturing test, tie to ground for normal operation.	
<b>FRAME#</b>	<b>Bidirectional</b>
Cycle Frame – is driven by the Universe when it is PCI initiator, and is monitored by the Universe when it is PCI target	
<b>GNT#</b>	<b>Input</b>
PCI Grant – indicates to the Universe that it has been granted ownership of the PCI bus.	
<b>IDSEL</b>	<b>Input</b>
PCI Initialization Device Select – is used as a chip select during configuration read and write transactions	
<b>LINT# [7:0]</b>	<b>Bidirectional (Open Drain)</b>
PCI Interrupt Inputs – these PCI interrupt inputs can be mapped to any PCI bus or VMEbus interrupt output.	
<b>IRDY#</b>	<b>Bidirectional</b>
Initiator Ready – is used by the Universe as PCI master to indicate that is ready to complete a current data phase.	
<b>LCLK</b>	<b>Input</b>
PCI Clock – provides timing for all transactions on the PCI bus. PCI signals are sampled on the rising edge of CLK, and all timing parameters are defined relative to this signal.	
<b>LOCK#</b>	<b>Bidirectional</b>
Lock – used by the Universe to indicate an exclusive operation with a PCI device. While the Universe drives LOCK#, other PCI masters are excluded from accessing that particular PCI device. Likewise, when the Universe samples LOCK#, it may be excluded from a particular PCI device.	

## 3.2 PCI Bus Signals (Continued)

<b>LRST#</b>	<b>Output</b>
PCI Reset Output – used to reset PCI resources.	
<b>PAR</b>	<b>Bidirectional</b>
Parity – parity is even across AD [31:0] and C/BE [3:0] (the number of 1s summed across these lines and PAR equal an even number).	
<b>PAR64</b>	<b>Bidirectional</b>
Parity Upper DWORD – parity is even across AD [63:32] and C/BE [7:4] (the number of 1s summed across these lines and PAR equal an even number).	
<b>PERR#</b>	<b>Bidirectional</b>
Parity Error – reports parity errors during all transactions. The Universe drives PERR# high within two clocks of receiving a parity error on incoming data, and holds PERR# for at least one clock for each errored data phase.	
<b>PLL_TESTOUT</b>	<b>Output</b>
Manufacturing Test Output—No connect	
<b>PLL_TESTSEL</b>	<b>Input</b>
Manufacturing Test Select—tie to ground for normal operation	
<b>PWRRST#</b>	<b>Input</b>
Power-up Reset – all Universe circuitry is reset by this input.	
<b>REQ#</b>	<b>Output</b>
Bus Request – used by the Universe to indicate that it requires the use of the PCI bus.	
<b>REQ64#</b>	<b>Bidirectional</b>
64-Bit Bus Request– used to request a 64-bit PCI transaction. If the target does not respond with ACK64#, 32-bit operation is assumed.	
<b>RST#</b>	<b>Input</b>
PCI Reset Input—resets the Universe from the PCI bus.	
<b>SERR#</b>	<b>Bidirectional</b>
System Error – reports address parity errors or any other system error.	
<b>STOP#</b>	<b>Bidirectional</b>
Stop – used by the Universe as PCI slave when it wishes to signal the PCI master to stop the current transaction. As PCI master, the Universe will terminate the transaction if it receives STOP# from the PCI slave.	

## 3.2 PCI Bus Signals (Continued)

<b>TCK</b>	<b>Input</b>
JTAG Test Clock Input – used to clock the Universe TAP controller. Tie to any logic level if JTAG is not used in the system.	
<b>TDI</b>	<b>Input</b>
JTAG Test Data Input – used to serially shift test data and test instructions into the Universe. Tie to any logic level if JTAG is not used in the system.	
<b>TDO</b>	<b>Output</b>
JTAG Test Data Output – used to serially shift test data and test instructions out of the Universe	
<b>TMODE [2:0]</b>	<b>Input</b>
Test Mode Enable – used for chip testing, tie to ground for normal operation.	
<b>TMS</b>	<b>Input</b>
JTAG Test Mode Select – controls the state of the Test Access Port (TAP) controller in the Universe. Tie to any logic level if JTAG is not used in the system.	
<b>TRDY#</b>	<b>Bidirectional</b>
Target Ready – used by the Universe as PCI slave to indicate that it is ready to complete the current data phase. During a read with Universe as PCI master, the slave asserts TRDY# to indicate to the Universe that valid data is present on the data bus.	
<b>TRST#</b>	<b>Input</b>
JTAG Test Reset – provides asynchronous initialization of the TAP controller in the Universe. Tie to ground if JTAG is not used in the system.	
<b>VCOCTL</b>	<b>Input</b>
Manufacturing testing, tie to ground for normal operation	
<b>VME_RESET#</b>	<b>Input</b>
VMEbus Reset Input — generates a VME bus system reset.	





## 4 Signals and DC Characteristics

### 4.1 Terminology

The I/O type abbreviations used in the pin list in Table 4.2 on page 4-3 are defined below. A numbered suffix indicates the current rating of the output (in mA).

Analog	Analog input signal
I	Input only
I/O	Input and output
O	Output only
OD	Open drain output
PD	Pulled-down internally
PU	Pulled-up internally
TP	Totem pole output
TTL	Input with TTL thresholds
TTL SCH	Schmitt trigger input with TTL thresholds
3S	Tri-state output

## 4.2 DC Characteristics and Pin Assignments

**Table 4.1 DC Electrical Characteristics ( $V_{DD} = 5\text{ V} \pm 10\%$ )**

Symbol	Parameter	Signal Type	Test Conditions	Tested at 0°C to 70°C	
				Min	Max
$V_{IH}$	Min. high-level input	CTTL	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	2.2 V	$V_{DD} + 0.3\text{V}$
		CMOS	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	$0.7V_{DD}$	$V_{DD} + 0.3\text{V}$
$V_{IL}$	Max. low-level input	CTTL	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	-0.3 V	0.8V
		CMOS	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	-0.3 V	$0.3V_{DD}$
$V_{T+}$	Positive going Schmitt trigger voltage	CTTL/SCH	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	–	2.4 V
		CMOS/SCH	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	–	$0.7V_{DD}$
$V_{T-}$	Negative going Schmitt trigger voltage	CTTL/SCH	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	0.8 V	–
		CMOS/SCH	$V_{OUT} = 0.1\text{V}$ or $V_{DD} - 0.1\text{V}$ ; $[I_{OUT}] = 20\text{ }\mu\text{A}$	$0.25V_{DD}$	–
$V_{Hysteresis}$	Schmitt trigger hysteresis voltage	CTTL/SCH	$V_{T+}$ to $V_{T-}$	$0.05V_{DD}$	–
		CMOS/SCH	$V_{T+}$ to $V_{T-}$	$0.12V_{DD}$	–
$I_{IN}$	Maximum input leakage current	CMOS and CTTL	With no pull-up resistor ( $V_{IN} = V_{SS}$ or $V_{DD}$ )	-5.0 $\mu\text{A}$	5.0 $\mu\text{A}$
$I_{OZ}$	Maximum output leakage current	3S	( $V_{OUT} = V_{SS}$ or $V_{DD}$ )	-10.0 $\mu\text{A}$	10.0 $\mu\text{A}$
		OD	( $V_{OUT} = V_{DD}$ )	-10.0 $\mu\text{A}$	10.0 $\mu\text{A}$

**Table 4.2 Pin List and DC Characteristics for Universe Signals<sup>1</sup>.**

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I <sub>OL</sub> (mA)	I <sub>OH</sub> (mA)	Signal Description
ACK64#	W11	W8	I/O	TTL	3S	6	–2	PCI Acknowledge 64 Bit Transfer
AD [63:0]	TABLE 4.3		I/O	TTL	3S	6	–2	PCI Address/Data Pins
C/BE# [0]	Y14	T11	I/O	TTL	3S	6	–2	PCI Command and Byte Enables
C/BE# [1]	V14	Y11						
C/BE# [2]	T14	U11						
C/BE# [3]	W13	W10						
C/BE# [4]	AE15	Y12						
C/BE# [5]	AD14	V11						
C/BE# [6]	T12	V10						
C/BE# [7]	AD12	Y9						
CLK64	C23	D16	I	TTL	–	–	–	VME Clock 64 MHz—60-40 duty, 5 ns rise time
DEVSEL#	AC7	V6	I/O	–	3S	6	–2	PCI Device Select
ENID	AE21	Y16	I	CMOS	–	–	–	Enable IDD Tests
FRAME#	W17	T12	I/O	TTL	3S	6	–2	PCI Cycle Frame
GNT#	AE17	Y14	I	TTL	–	–	–	PCI Grant
IDSEL	AB16	Y15	I	TTL	–	–	–	PCI Initialization Device Select
LINT# [0]	K20	H15	I/O	TTL	OD	12	–12	PCI Interrupt
LINT# [1]	AA5	U3						
LINT# [2]	L9	J3						
LINT# [3]	V6	R4						
LINT# [4]	M4	J4						
LINT# [5]	L3	J6						
LINT# [6]	M8	J5						
LINT# [7]	L1	K4						
IRDY#	AC15	V12	I/O	TTL	3S	6	–2	PCI Initiator Ready
LCLK	AA3	W3	I	TTL	–	–	–	PCI Clock Signal
LOCK#	AA23	T18	I/O	TTL	3S	6	–2	PCI Lock
LRST#	R1	M1	O	–	3S	6	–2	PCI Reset Output
PAR	P8	L1	I/O	TTL	3S	6	–2	PCI parity
PAR64	AE5	W4	I/O	TTL	3S	6	–2	PCI Parity Upper DWORD
PERR#	AB4	W5	I/O	TTL	3S	6	–2	PCI Parity Error
PLL_TESTOUT	AB2	V1	FOR FACTORY TESTING					

**Table 4.2 Pin List and DC Characteristics for Universe Signals<sup>1</sup>. (Continued)**

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I <sub>OL</sub> (mA)	I <sub>OH</sub> (mA)	Signal Description
PLL_TESTSEL	AC1	T2	FOR FACTORY TESTING					
PWRRST#	T4	R1	I	TTL/SCHM	–	–	–	Power-up Reset
REQ#	K22	F20	O	–	3S	6	–2	PCI Request
REQ64#	AD18	T13	I/O	TTL	3S	6	–2	PCI Request 64 Bit Transfer
RST#	AA19	W17	I	TTL	–	–	–	PCI Reset
SERR#	AA7	R7	O	TTL	OD	12	–12	PCI System Error
STOP#	AB18	W15	I/O	TTL	3S	6	–2	PCI Stop
TCK	H12	A10	I	TTL	–	–	–	JTAG Test Clock Input
TDI	A13	F10	I	TTL (PU)	–	–	–	JTAG Test Data Input
TDO	C13	E11	O	–	3S	–	–	JTAG Test Data OUTput
TMODE [0]	AA13	W11	I	TTL	–	–	–	Test Mode Enable
TMODE [1]	AA21	V17						
TMODE [2]	W23	R18						
TMS	C11	C9	I	TTL (PU)	–	–	–	JTAG Test Mode Select
TRDY#	AD8	W7	I/O	TTL	3S	6	–2	PCI Target Ready
TRST#	E13	B10	I	TTL (PU)	–	–	–	JTAG Test Reset
VA [31:0]	TABLE 4.4		I/O	TTL (PD)	3S	3	–3	VMEbus Address Pins
VAM [0]	E11	D8	I/O	TTL	3S	3	–3	VMEbus Address Modifier Signals
VAM [1]	D10	A6						
VAM [2]	G9	E9						
VAM [3]	B10	B8						
VAM [4]	H10	D9						
VAM [5]	A9	A7						
VAM_DIR	B8	E8	O	–	3S	6	–6	VMEbus AM Signal Direction Control
VAS#	B14	A12	I/O	TTL/SCHM (PU)	3S	3	–3	VMEbus Address Strobe
VAS_DIR	K12	D10	O	–	3S	6	–6	VMEbus AS Direction Control
VA_DIR	G13	B11	O	–	3S	12	–12	VMEbus Address Direction Control

**Table 4.2 Pin List and DC Characteristics for Universe Signals<sup>1</sup>. (Continued)**

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I <sub>OL</sub> (mA)	I <sub>OH</sub> (mA)	Signal Description
VBCLR#	N3	K5	O	—	3S	3	−3	VMEbus BCLR* Signal
VBGI# [0]	N21	K19	I	TT	—	—	—	VMEbus Bus Grant In
VBGI# [1]	M16	K17						
VBGI# [2]	N25	K15		TT (PD)				
VBGI# [3]	N23	L16						
VBGO# [0]	M20	K16	O	—	3S	12	−12	VMEbus Bus Grant Out
VBGO# [1]	L25	J20						
VBGO# [2]	M18	K20						
VBGO# [3]	M24	K18						
VCOCTL	AE3	T5	I	—	—	—	—	Factory testing
VD [31:0]	TABLE 4.4		I/O	TTL	3S	3	−3	VMEbus Data Pins
VD_DIR	F10	F8	O	—	3S	12	−12	VMEbus Data Direction Control
VDS# [0]	F12	E10	I/O	TTL (PU)	3S	3	−3	VMEbus Data Strobes
VDS# [1]	A11	A9						
VDS_DIR	J11	F9	O	—	3S	6	−6	VMEbus Data Strobe Direction Control
VDTACK#	G15	B13	I/O	TTL/ SCHM (PU)	3S	3	−3	VMEbus DTACK* Signal
VIACK#	E7	E6	I/O	TTL	3S	3	−3	VMEbus IACK* Signal
VIACKI#	AE23	W16	I	TTL	—	—	—	VMEbus IACKIN* Signal
VIACKO#	L21	H17	O	—	3S	12	−12	VMEbus IACKOUT* Signal
VLWORD#	K14	C11	I/O	TTL (PD)	3S	3	−3	VMEbus LWORD* Signal
VME_RESET#	V22	T19	I	TTL				VMEbus Reset Input
VOE#	B12	C10	O	—	3S	24	−24	VMEbus Transceiver Output Enable
VRACFAIL#	P18	M16	I	TTL/ SCHM	—	—	—	VMEbus ACFAIL* Signal
VRBBSY#	M6	J2	I	TTL/ SCHM	—	—	—	VMEbus Received BBSY* Signal
VRBERR#	A7	B7	I	TTL/ SCHM	—	—	—	VMEbus Receive Bus Error

**Table 4.2 Pin List and DC Characteristics for Universe Signals<sup>1</sup>. (Continued)**

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I <sub>OL</sub> (mA)	I <sub>OH</sub> (mA)	Signal Description
VRBR# [0]	W5	U2	I	TTL/ SCHM	–	–	–	VMEbus Receive Bus Request
VRBR# [1]	U1	P1						
VRBR# [2]	R3	M3						
VRBR# [3]	L7	H2						
VRIRQ# [1]	H22	F19	I	TTL/ SCHM	–	–	–	VMEbus Receive Interrupts
VRIRQ# [2]	H20	F17						
VRIRQ# [3]	E25	E20						
VRIRQ# [4]	J21	G18						
VRIRQ# [5]	V16	U12						
VRIRQ# [6]	P20	M19						
VRIRQ# [7]	R17	M18						
VRSYSFAIL#	AC13	T10	I	TTL	–	–	–	VMEbus Receive SYSFAIL Signal
VRSYSRST#	C21	C16	I	TTL/ SCHM	–	–	–	VMEbus Receive SYSRESET* Signal
VSCON_DIR	M2	J1	O	–	3S	6	–6	SYSCON signals direction control
VSLAVE_DIR	C15	F12	O	–	3S	6	–6	DTACK/BERR direction control
VSYCLK	N7	K2	I/O	TTL	3S	3	–3	VMEbus SYCLK Signal
VWRITE#	D8	B6	I/O	TTL	3S	3	–3	VMEbus Write
VXBBSY	P2	L3	O	–	3S	3	–3	VMEbus Transmit BBSY* Signal
VXBERR	D12	B9	O	–	3S	3	–3	VMEbus Transmit Bus Error (BERR*)
VXBR [0]	G25	G19	O	–	3S	3	–3	VMEbus Transmit Bus Request
VXBR [1]	H24	H16						
VXBR [2]	P24	M20						
VXBR [3]	G23	C19						

**Table 4.2 Pin List and DC Characteristics for Universe Signals<sup>1</sup>.** (Continued)

Pin Name	PBGA Pin Number	CBGA Pin Number	Type	Input Type	Output Type	I <sub>OL</sub> (mA)	I <sub>OH</sub> (mA)	Signal Description
VXIRQ [1]	J19	J16	O	–	3S	3	–3	VMEbus Transmit Interrupts
VXIRQ [2]	K24	H19						
VXIRQ [3]	K18	J17						
VXIRQ [4]	J25	G20						
VXIRQ [5]	L23	J18						
VXIRQ [6]	M22	J19						
VXIRQ [7]	R25	L17						
VXSYSFAIL	M10	K3	O	–	3S	3	–3	VMEbus Transmit SYSFAIL Signal
VXSYSRST	A23	E16	O	–	3S	3	–3	VMEbus Transmit SYSRESET* Signal

Note 1: All PCI pins meet PCI's AC current specifications.

**Table 4.3 PCI Bus Address/Data Pins**

Signal	PBGA	CBGA	Signal	PBGA	CBGA
AD [0]	P16	L18	AD [32]	L17	J15
AD [1]	P22	M17	AD [33]	N19	L19
AD [2]	R19	N19	AD [34]	R23	M15
AD [3]	T18	U20	AD [35]	U19	N18
AD [4]	T22	N15	AD [36]	U23	R20
AD [5]	T20	T20	AD [37]	W25	P16
AD [6]	AA25	U19	AD [38]	U21	P17
AD [7]	AB24	R17	AD [39]	V20	R19
AD [8]	AB22	R16	AD [40]	Y22	U18
AD [9]	AE25	T17	AD [41]	W21	P15
AD [10]	AC21	V19	AD [42]	AD22	Y18
AD [11]	AB20	V15	AD [43]	Y20	T15
AD [12]	AC19	W18	AD [44]	AD20	T14
AD [13]	AA17	V14	AD [45]	Y18	U15
AD [14]	AA15	U13	AD [46]	AE19	W14
AD [15]	U15	R12	AD [47]	AD16	W13
AD [16]	AE11	U10	AD [48]	V12	T9

**Table 4.3 PCI Bus Address/Data Pins**

Signal	PBGA	CBGA		Signal	PBGA	CBGA
AD [17]	AB12	U9		AD [49]	Y12	W9
AD [18]	W9	V8		AD [50]	AC11	R9
AD [19]	AD10	U8		AD [51]	V10	Y4
AD [20]	AE7	T7		AD [52]	AB10	R8
AD [21]	Y8	W6		AD [53]	AA9	U7
AD [22]	AD4	U6		AD [54]	AB8	T6
AD [23]	Y6	R5		AD [55]	AB6	V4
AD [24]	Y2	P5		AD [56]	Y4	R3
AD [25]	V4	R2		AD [57]	W3	V2
AD [26]	U5	P3		AD [58]	AA1	T1
AD [27]	W1	P2		AD [59]	V2	N5
AD [28]	U7	M5		AD [60]	R5	N4
AD [29]	T8	M4		AD [61]	T2	N2
AD [30]	P6	L5		AD [62]	R9	M6
AD [31]	P10	L4		AD [63]	N5	L2



**Table 4.4 VMEbus Address Pins<sup>a</sup>**

Signal	PBGA	CBGA		Signal	PBGA	CBGA
VA [1]	H14	E12		VA [17]	D18	B16
VA [2]	A15	D11		VA [18]	C19	C15
VA [3]	F14	B12		VA [19]	B20	D17
VA [4]	J15	C12		VA [20]	B22	D15
VA [5]	D14	D12		VA [21]	D20	C17
VA [6]	G17	C13		VA [22]	F20	E15
VA [7]	H16	A17		VA [23]	E19	F14
VA [8]	B16	D13		VA [24]	A25	C20
VA [9]	C17	A15		VA [25]	E23	B18
VA [10]	D16	F13		VA [26]	C25	E19
VA [11]	A19	E14		VA [27]	G21	F16
VA [12]	B18	B14		VA [28]	E21	D18
VA [13]	F16	A16		VA [29]	F22	F18
VA [14]	E17	D14		VA [30]	D24	D19
VA [15]	A21	B17		VA [31]	F24	G16
VA [16]	F18	B15				

*a. All VA pins have an internal pull-down.*

**Table 4.5 VMEbus Data Pins<sup>a</sup>**

Signal	PBGA	CBGA		Signal	PBGA	CBGA
VD [0]	J7	H3		VD [16]	F6	E2
VD [1]	K8	D1		VD [17]	G5	G6
VD [2]	K2	H4		VD [18]	B2	E5
VD [3]	J3	F1		VD [19]	C1	E3
VD [4]	K4	H6		VD [20]	E3	E4
VD [5]	G1	G5		VD [21]	C3	A3
VD [6]	H2	G2		VD [22]	A1	C2
VD [7]	K6	E1		VD [23]	A3	B5
VD [8]	J5	G4		VD [24]	C5	C4
VD [9]	E1	D2		VD [25]	D6	C6
VD [10]	H6	F2		VD [26]	B4	B4
VD [11]	H4	F5		VD [27]	B6	E7
VD [12]	G3	F3		VD [28]	C7	B3
VD [13]	F2	D4		VD [29]	F8	D6
VD [14]	D2	F4		VD [30]	A5	A5
VD [15]	F4	D3		VD [31]	E9	C7

*a. VD[30:27] have internal pull-downs.*

**Table 4.6 Pin Assignments for Power and Ground**

<b>V<sub>SS</sub> Pins</b>		<b>V<sub>DD</sub> Pins</b>		
<b>PBGA</b>		<b>PBGA</b>		
AB14	N15	A17	H8	W15
AC9	N17	AA11	H18	W19
AC25	P4	AC3	J1	Y24
AD6	P12	*AC5	J9	
*AE1	P14	AC17	J17	
AE13	R11	AC23	J23	
J13	R13	AD2	L5	
K10	R15	AD24	L19	
K16	T6	AE9	R7	
L11	T16	B24	R21	
L13	T24	C9	T10	
L15	U11	D4	U3	
M12	U13	D22	U9	
M14	V24	E5	U17	
N1	Y10	E15	U25	
N9	Y16	G7	V8	
N11		G11	V18	
N13		G19	W7	

\*AVDD and AVSS are power pins specifically used for powering the analog circuitry in the Universe. Extra care should be taken to avoid noise and ground shifting on these pins through the use of decoupling capacitors or isolated ground and power planes.

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	T	U	V	W	Y	AA	AB	AC	AD	AE		
1	VDI22		VDI19		VDI9		VDI5		VDD		INT#7		VSS		VRBR#1		VRBR#1		AD27		AD18		PLL_TESTSEL		AVSS		
2		VDI18		VDI14		VDI13		VDI6		VDI2		VSCON_DIR		VXBBSY		AD161		AD159		AD124		PLL_TESTOUT		VDD			
3	VDI23		VDI21		VDI20		VDI12		VDI3		INT#5		VBCLR#		VRBR#2		VDD		AD157		LCCLK		VDD		VCCOCTL		
4		VDI26		VDD		VDI15		VDI11		VDI4		INT#4		VSS		PWRERST#		AD125		AD156		PERR#		AD122			
5	VDI30		VDI24		VDD		VDI17		VDI8		VDD		AD163		AD160		AD126		VRBR#0		INT#11		AVDD		PAR64		
6		VDI27		VDI25		VDI16		VDI10		VDI7		VRBBSY#		AD130		VSS		INT#3		AD123		AD155		VSS			
7	VRBERR#		VDI28		VIACK#		VDD		VDI0		VRBR#3		VSSYCLK		VDD		AD128		VDD		SERR#		DENSEL#		AD120		
8		VAM_DIR		VWRITE#		VDI29		VAM2		VDI1		INT#6		PAR		AD162		VDD		AD118		AD154		TRDY#			
9	VAM5		VDD		VDI31		VAM2		VDD		INT#2		VSS		AD162		VDD		AD158		AD153		VSS		VDD		
10		VAM3		VAM11		VD_DIR		VAM4		VSS		VXSYSFAIL		AD131		VDD		AD151		VSS		AD152					
11	VDS#11		TMS		VAM0		VDD		VDS_DIR		VSS		VSS		VSS		VSS		ACK6#		VDD		AD150		AD116		
12		VOE#		VXBERR		VDS#0		TCK		VAS_DIR		VSS		VSS		CBE6		AD148		AD149		AD171		CBE7			
13	TDI		TDO		TRST#		VA_DIR		VSS		VSS		VSS		VSS		VSS		CBE13		TMODE0		VRSYS_FAIL#		VSS		
14		VAS#		VA5		VA13		VA11		VLWORD#		VSS		VSS		CBE2		CBE11		CBE0		VSS		CBE5			
15	VA12		VSLAVE_DIR		VDD		VDTACK#		VA14		VSS		VSS		VSS		AD115		VDD		AD14		IRDY#		CBE4		
16		VA18		VA100		VA113		VA17		VSS		VBGH#1		AD00		VSS		VRBRQ#5		VSS		IDSEL		AD147			
17	VDD		VA9		VA144		VA16		VDD		AD132		VSS		VRBRQ#7		VDD		FRAME#		AD113		VDD		GNT#		
18		VA112		VA117		VA116		VDD		VXBRQ3		VBGO#2		VRACFAIL#		AD13		VDD		AD145		STOP#		REQ6#			
19	VA111		VA118		VA123		VDD		VXBRQ11		VDD		AD133		AD12		AD135		VDD		RST#		AD12		AD146		
20		VA119		VA121		VA122		VRBRQ#2		INT#0		VBGO#0		VRBRQ#6		AD15		AD139		AD143		AD111		AD144			
21	VA115		VRSYSRST#		VA128		VA127		VRBRQ#4		VIACK#		VBGH#0		VDD		AD138		AD141		TMODE11		AD100		END		
22		VA120		VDD		VA129		VRBRQ#1		REQ#		VXBRQ6		AD11		AD14		VME_RST#		AD140		AD18		AD142			
23	VXSYSRST		CLK64		VA125		VXBRE3		VDD		VXBRQ5		VBGH#3		AD134		AD156		TMODE12				VDD		VIACK#		
24		VDD		VA130		VA131		VXBRE1		VXBRQ2		VBGO#3		VXBRE2		VSS		VSS		VDD		AD17		VDD			
25	VA124		VA126		VRBRQ#3		VXBRE0		VXBRQ4		VBGO#1		VBGH#2		VXBRQ7		VDD		AD137		AD16		VSS		AD19		

Figure 4. 1: Pinout for 313-pin Plastic BGA Package

(Top view)

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R	T	U	V	W	Y
1			VSS	VD[1]	VD[7]	VD[3]	VSS	VDD	VSCON_DIR	VDD	PAR	LRST#	VDD	VRBR#[1]	PWRBST#	AD[58]	VDD	PLL_TESTOUT		
2		VSS	VD[22]	VD[9]	VD[6]	VD[10]	VD[6]	VRBR#[3]	VRBBSY#	VSSCLK	AD[63]	VSS	AD[61]	AD[7]	AD[25]	PLL_TESTSEL	VRBR#[0]	AD[57]	VSS	
3	VD[21]	VD[28]	VDD	VD[15]	VD[9]	VD[12]	VDD	VD[0]	INT#[2]	VSSYSFALL	VXBBSY	VRBR#[2]	VDD	AD[56]	AD[56]	VDD	INT#[11]	VDD	LCCLK	VSS
4	VDD	VD[26]	VD[24]	VD[13]	VD[20]	VD[14]	VD[8]	VD[2]	INT#[4]	INT#[7]	AD[31]	AD[29]	AD[60]	VDD	INT#[3]	VDD	VSS	AD[55]	PAR64	AD[51]
5	VD[30]	VD[23]	VDD	VDD	VD[18]	VD[11]	VD[5]	VDD	INT#[6]	VBCLR#	AD[30]	AD[38]	AD[59]	AD[24]	AD[23]	VCOCTL	AVSS	AVDD	PERR#	VSS
6	VAM[1]	VWRIT#	VD[25]	VD[29]	VIACK#	VDD	VD[17]	VD[4]	INT#[5]	VSS	VSS	AD[62]	VSS	VSS	VDD	AD[54]	AD[22]	DEVSEL#	AD[21]	VSS
7	VAM[5]	VRBR#	VD[31]	VDD	VD[27]	VSS									SER#	AD[20]	AD[53]	VDD	TRDY#	VSS
8	VDD	VAM[3]	VDD	VAM[0]	VAM_DIR	VD_DIR									AD[52]	VDD	AD[19]	AD[15]	ACK64#	VDD
9	VDS#	VXBERR	TMS	VAM[4]	VAM[2]	VDS_DIR									AD[50]	AD[48]	AD[17]	VSS	AD[49]	CBH[7]
10	TCK	TEST#	VOE#	VAS_DIR	VDS#	TDI									VSS	VRSYSFALL#	AD[16]	CBH[6]	CBH[3]	VDD
11	VDD	VA_DIR	VLWORD#	VA[2]	TDO	VSS									VSS	CBH[0]	CBH[2]	CBH[5]	TMODE[0]	CBH[1]
12	VAS#	VA[3]	VA[4]	VA[5]	VA[1]	VSLAVE_DIR									AD[15]	FRAME#	VRIRO#	IRDY#	VSS	CBH[4]
13	VDD	VDTACK#	VA[6]	VA[8]	VDD	VA[10]									VSS	REQ64#	AD[14]	VDD	AD[47]	VDD
14	VSS	VA[12]	VDD	VA[14]	VA[11]	VA[23]									VSS	AD[44]	VDD	AD[13]	AD[46]	GNT#
15	VA[9]	VA[16]	VA[18]	VA[20]	VA[22]	VDD	VSS	INT#	AD[32]	VBGH#	VSS	AD[34]	AD[4]	AD[41]	VDD	AD[43]	AD[45]	AD[11]	STOP#	IDSEL
16	VA[13]	VA[17]	VRSYSRST#	CLK64	VXSYSRST	VA[27]	VA[31]	VXBR[1]	VXBRQ[1]	VBGO#	VBGH#	VRACTAIL#	VDD	AD[37]	AD[8]	VSS	VDD	VDD	VIACK#	END
17	VA[7]	VA[15]	VA[21]	VA[19]	VDD	VRBRQ#	VDD	VIACK#	VXBRQ[3]	VBGH#	VXBRQ[7]	AD[1]	VSS	AD[38]	AD[7]	AD[9]	VDD	TMODE[1]	RST#	VDD
18	VSS	VA[25]	VDD	VA[28]	VDD	VA[29]	VRBRQ#	VDD	VXBRQ[5]	VBGO#	AD[0]	VRBRQ#	AD[35]	VDD	TMODE[2]	LOCK#	AD[40]	VDD	AD[12]	AD[42]
19		VXBR[3]	VXBR[3]	VA[30]	VA[26]	VRBRQ#	VXBR[0]	VXBRQ[2]	VXBRQ[6]	VBGH#	AD[33]	VRBRQ#	AD[2]	VSS	AD[39]	VME_RESET#	AD[6]	AD[10]	VSS	
20			VA[24]	VDD	VRBRQ[3]	REQ#	VXBRQ[4]	VDD	VBGO#	VBGO#	VDD	VXBR[2]	VDD	VSS	AD[36]	AD[5]	AD[3]	VSS		

Figure 4. 2: Pinout for 324-pin Ceramic BGA Package

(Top view)



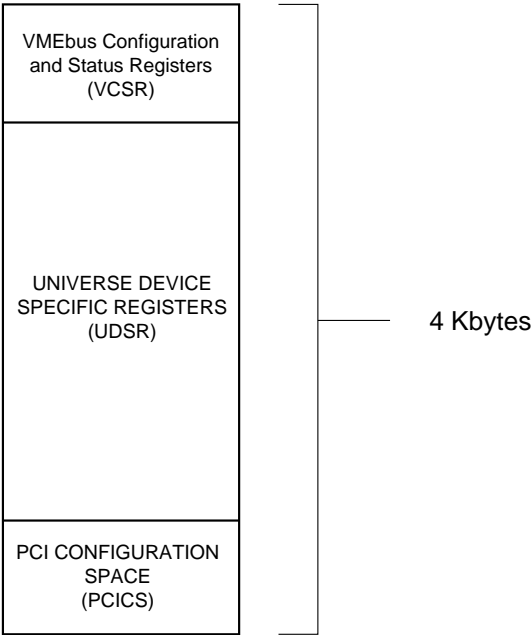
# Appendix A Registers

The Universe Control and Status Registers (UCSR) facilitate host system configuration and allow the user to control Universe operational characteristics. The UCSRs are divided into three groups:

- PCI Configuration Space (PCICS),
- VMEbus Control and Status Registers (VCSR), and
- Universe Device Specific Status Registers (UDSR).

The Universe registers are little-endian.

Figure A.1 below summarizes the supported register access mechanisms.



**Figure A.1 : UCSR Access Mechanisms**

Table A.1 below lists the Universe registers by address offset. The tables following the register map (Table A.2 to Table A.84) provide detailed descriptions of each register.

Address offsets in Table A.1 below apply to accesses from the PCI bus and to accesses from the VMEbus side using the VMEbus Register Access Image (see “Registers” on page 2-87). For register accesses in CR/CSR space, be sure to add 508 Kbytes (0x7F000) to the address offsets provided in the table.

The resets that apply to the registers are provided on a bit by bit basis. If "all" is given in the "Reset By" field in the register description, then that bit is reset by power reset (PWRRST#), the VMEbus SYSRST\* input (VRSYSRST#), and the PCI RST# input. If "PWR, VME" are given in the "Reset By" field in the register description, then that bit is reset by power reset (PWRRST#) and the VMEbus SYSRST\* input (VRSYSRST#). Bits not affected by resets contain "none" in the "Reset By" field in the register description.

**Table A.1 : Universe Register Map**

Offset	Register	Name
000	PCI Configuration Space ID Register	PCI_ID
004	PCI Configuration Space Control and Status Register	PCI_CSR
008	PCI Configuration Class Register	PCI_CLASS
00C	PCI Configuration Miscellaneous 0 Register	PCI_MISC0
010	PCI Configuration Base Address Register	PCI_BS
014	PCI Unimplemented	
018	PCI Unimplemented	
01C	PCI Unimplemented	
020	PCI Unimplemented	
024	PCI Unimplemented	
028	PCI Reserved	
02C	PCI Reserved	
030	PCI Unimplemented	
034	PCI Reserved	
038	PCI Reserved	
03C	PCI Configuration Miscellaneous 1 Register	PCI_MISC1
040-0FF	PCI Unimplemented	
100	PCI Slave Image 0 Control	LSI0_CTL
104	PCI Slave Image 0 Base Address Register	LSI0_BS
108	PCI Slave Image 0 Bound Address Register	LSI0_BD
10C	PCI Slave Image 0 Translation Offset	LSI0_TO
110	Universe Reserved	



**Table A.1 : Universe Register Map**

Offset	Register	Name
114	PCI Slave Image 1 Control	LSI1_CTL
118	PCI Slave Image 1 Base Address Register	LSI1_BS
11C	PCI Slave Image 1 Bound Address Register	LSI1_BD
120	PCI Slave Image 1 Translation Offset	LSI1_TO
124	Universe Reserved	
128	PCI Slave Image 2 Control	LSI2_CTL
12C	PCI Slave Image 2 Base Address Register	LSI2_BS
130	PCI Slave Image 2 Bound Address Register	LSI2_BD
134	PCI Slave Image 2 Translation Offset	LSI2_TO
138	Universe Reserved	
13C	PCI Slave Image 3 Control	LSI3_CTL
140	PCI Slave Image 3 Base Address Register	LSI3_BS
144	PCI Slave Image 3 Bound Address Register	LSI3_BD
148	PCI Slave Image 3 Translation Offset	LSI3_TO
14C-16C	Universe reserved	
170	Special Cycle Control Register	SCYC_CTL
174	Special Cycle PCI bus Address Register	SCYC_ADDR
178	Special Cycle Swap/Compare Enable Register	SCYC_EN
17C	Special Cycle Compare Data Register	SCYC_CMP
180	Special Cycle Swap Data Register	SCYC_SWP
184	PCI Miscellaneous Register	LMISC
188	Special PCI Slave Image	SLSI
18C	PCI Command Error Log Register	L_CMDERR
190	PCI Address Error Log	LAERR
194-1FC	Universe reserved	
200	DMA Transfer Control Register	DCTL
204	DMA Transfer Byte Count Register	DTBC
208	DMA PCI bus Address Register	DLA
20C	Universe Reserved	
210	DMA VMEbus Address Register	DVA
214	Universe Reserved	
218	DMA Command Packet Pointer	DCPP
21C	Universe Reserved	
220	DMA General Control and Status Register	DGCS
224	DMA Linked List Update Enable Register	D_LLUE
228-2FC	Universe reserved	
300	PCI Interrupt Enable	LINT_EN

**Table A.1 : Universe Register Map**

Offset	Register	Name
304	PCI Interrupt Status	LINT_STAT
308	PCI Interrupt Map 0	LINT_MAP0
30C	PCI Interrupt Map 1	LINT_MAP1
310	VMEbus Interrupt Enable	VINT_EN
314	VMEbus Interrupt Status	VINT_STAT
318	VMEbus Interrupt Map 0	VINT_MAP0
31C	VMEbus Interrupt Map 1	VINT_MAP1
320	Interrupt Status/ID Out	STATID
324	VIRQ1 STATUS/ID	V1_STATID
328	VIRQ2 STATUS/ID	V2_STATID
32C	VIRQ3 STATUS/ID	V3_STATID
330	VIRQ4 STATUS/ID	V4_STATID
334	VIRQ5 STATUS/ID	V5_STATID
338	VIRQ6 STATUS/ID	V6_STATID
33C	VIRQ7 STATUS/ID	V7_STATID
340-3FC	Universe reserved	
400	Master Control	MAST_CTL
404	Miscellaneous Control	MISC_CTL
408	Miscellaneous Status	MISC_STAT
40C	User AM Codes Register	USER_AM
410-EFC	Universe reserved	
F00	VMEbus Slave Image 0 Control	VSI0_CTL
F04	VMEbus Slave Image 0 Base Address Register	VSI0_BS
F08	VMEbus Slave Image 0 Bound Address Register	VSI0_BD
F0C	VMEbus Slave Image 0 Translation Offset	VSI0_TO
F10	Universe reserved	
F14	VMEbus Slave Image 1 Control	VSI1_CTL
F18	VMEbus Slave Image 1 Base Address Register	VSI1_BS
F1C	VMEbus Slave Image 1 Bound Address Register	VSI1_BD
F20	VMEbus Slave Image 1 Translation Offset	VSI1_TO
F24	Universe reserved	
F28	VMEbus Slave Image 2 Control	VSI2_CTL
F2C	VMEbus Slave Image 2 Base Address Register	VSI2_BS
F30	VMEbus Slave Image 2 Bound Address Register	VSI2_BD
F34	VMEbus Slave Image 2 Translation Offset	VSI2_TO
F38	Universe reserved	
F3C	VMEbus Slave Image 3 Control	VSI3_CTL

**Table A.1 : Universe Register Map**

Offset	Register	Name
F40	VMEbus Slave Image 3 Base Address Register	VSI3_BS
F44	VMEbus Slave Image 3 Bound Address Register	VSI3_BD
F48	VMEbus Slave Image 3 Translation Offset	VSI3_TO
F4C-F6C	Universe reserved	
F70	VMEbus Register Access Image Control Register	VRAI_CTL
F74	VMEbus Register Access Image Base Address	VRAI_BS
F78	Universe reserved	
F7C	Universe reserved	
F80	VMEbus CSR Control Register	VCSR_CTL
F84	VMEbus CSR Translation Offset	VCSR_TO
F88	VMEbus AM Code Error Log	V_AMERR
F8C	VMEbus Address Error Log	VAERR
F90-FEC	Universe reserved	
FF0	VME CR/CSR Reserved	
FF4	VMEbus CSR Bit Clear Register	VCSR_CLR
FF8	VMEbus CSR Bit Set Register	VCSR_SET
FFC	VMEbus CSR Base Address Register	VCSR_BS



*Caution: Register space marked as “reserved” should not be overwritten. Unimplemented registers return a value of 0 on reads; writes complete normally.*

**Table A.2 : PCI Configuration Space ID Register (PCI\_ID)**

Register Name: PCI_ID		Offset:000
Bits	Function	
31-24	DID	
23-16	DID	
15-08	VID	
07-00	VID	

**PCI\_ID Description**

Name	Type	Reset By	Reset State	Function
DID[15:0]	R	all	0	Device ID - Newbridge allocated device identifier
VID[15:0]	R	all	10E3	Vendor ID - PCI SIG allocated vendor identifier

**Table A.3 : PCI Configuration Space Control and Status Register (PCI\_CSR)**

Register Name: PCI_CSR					Offset:004			
Bits	Function							
31-24	D_PE	S_SERR	R_MA	R_TA	S_TA	DEVSEL		DP_D
23-16	TFBBC	PCI Reserved						
15-08	PCI Reserved					MFBBC		SERR_EN
07-00	WAIT	PERESP	VGAPS	MWI_EN	SC	BM	MS	IOS

**PCI\_CSR Description**

Name	Type	Reset By	Reset State	Function
D_PE	R/Write 1 to Clear	all	0	Detected Parity Error 0=No parity error, 1=Parity error This bit is always set by the Universe when: the PCI master interface detects a data parity error or the PCI target interface detects address or data parity errors.
S_SERR	R/Write 1 to Clear	all	0	Signalled SERR# 0=SERR# not asserted, 1=SERR# asserted. The Universe PCI target interface sets this bit when it asserts SERR# to signal an address parity error. SERR_EN must be set before SERR# can be asserted.
R_MA	R/Write 1 to Clear	all	0	Received Master-Abort 0=Master did not generate Master-Abort, 1=Master generated Master-Abort The Universe PCI master interface sets this bit when a transaction it initiated had to be terminated with a Master-Abort.
R_TA	R/Write 1 to Clear	all	0	Received Target-Abort 0=Master did not detect Target-Abort, 1=Master detected Target-Abort. The Universe PCI master interface sets this bit when a transaction it initiated was terminated with a Target-Abort.
S_TA	R/Write 1 to Clear	all	0	Signalled Target-Abort 0=Target did not terminate transaction with Target-Abort, 1=Target terminated transaction with Target-Abort.
DEVSEL	R	all	01	Device Select Timing The Universe is a medium speed device
DP_D	R/Clear	all		Data Parity Detected 0=Master did not detect/generate data parity error, 1=Master detected/generated data parity error. The Universe PCI master interface sets this bit if the Parity Error Response bit is set, it is the master of transaction in which it asserts PERR#, or the addressed target asserts PERR#.
TFBBC	R	all	0	Target Fast Back to Back Capable Universe cannot accept Back to Back cycles from a different agent.

**PCI\_CSR Description**

<b>Name</b>	<b>Type</b>	<b>Reset By</b>	<b>Reset State</b>	<b>Function</b>
MFBBC	R	all	0	Master Fast Back to Back Enable The Universe master never generates fast back to back transactions.
SERR_EN	R/W	all	0	SERR# Enable 0=Disable SERR# driver, 1=Enable SERR# driver. Setting this and PERESP allows the Universe PCI target interface to report address parity errors with SERR#.
WAIT	R	all	0	Wait Cycle Control 0=No address/data stepping
PERESP	R/W	all	0	Parity Error Response 0=Disable, 1=Enable Controls the Universe response to data and address parity errors. When enabled, it allows the assertion of PERR# to report data parity errors. When this bit and SERR_EN are asserted, the Universe can report address parity errors on SERR#. Universe parity generation is unaffected by this bit.
VGAPS	R	all	0	VGA Palette Snoop 0=Disable
MWI_EN	R	all	0	Memory Write and Invalidate Enable 0=Disable The Universe PCI master interface never generates a Memory Write and Invalidate command.
SC	R	all	0	Special Cycles 0=Disable The Universe PCI target interface never responds to special cycles.
BM	R/W	PWR, VME	see note 1	Master Enable 0=Disable, 1=Enable For a VMEbus slave image to respond to an incoming cycle, this bit must be set.
MS	R/W	PWR, VME	see note 1	Target Memory Enable 0=Disable, 1=Enable
IOS	R/W	PWR, VME	see note 1	Target IO Enable 0=Disable, 1=Enable

Note:

1. These bits may not be disabled after reset as required by the PCI specification if the VCSR or LSI0 power-up options have been enabled.

**Table A.4 : PCI Configuration Class Register (PCI\_CLASS)**

Register Name: PCI_CLASS		Offset:008
Bits	Function	
31-24	BASE [7:0]	
23-16	SUB [7:0]	
15-08	PROG [7:0]	
07-00	RID [7:0]	

**PCI\_CLASS Description**

Name	Type	Reset By	Reset State	Function
BASE [7:0]	R	all	06	Base Class Code The Universe is defined as a PCI bridge device
SUB [7:0]	R	all	80	Sub Class Code The universe sub-class is "other bridge device"
PROG [7:0]	R	all	00	Programming Interface The Universe does not have a standardized register-level programming interface
RID [7:0]	R	all	00	Revision ID

**Table A.5 : PCI Configuration Miscellaneous 0 Register (PCI\_MISC0)**

Register Name: PCI_MISC0				Offset:00C			
Bits	Function						
31-24	BISTC	SBIST	PCI Reserved		CCODE		
23-16	MFUNCT	LAYOUT					
15-08	LTIMER				0	0	0
07-00	PCI Unimplemented						

**PCI\_MISC0 Description**

Name	Type	Reset By	Reset State	Function
BISTC	R	all	0	The Universe is not BIST Capable
SBIST	R	all	0	Start BIST The Universe is not BIST capable
CCODE	R	all	0	Completion Code The Universe is not BIST capable
MFUNCT	R	all	1	Multifunction Device 0=No, 1=Yes The Universe is a multifunction device with a single function: function 0.
LAYOUT	R	all	0	Configuration Space Layout
LTIMER [7:3]	R/W	all	0	Latency Timer: The latency timer has a resolution of 8 clocks



**Table A.6 : PCI Configuration Base Address Register (PCI\_BS)**

Register Name: PCI_BS					Offset:010			
Bits	Function							
31-24	BS							
23-16	BS							
15-08	0	0	0	0	0	0	0	0
07-00	0	0	0	0	0	0	0	SPACE

**PCI\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	all	0	Base Address
SPACE	R	all	Power-up Option	PCI bus Address Space 0=Memory, 1=I/O

This register specifies the 64 Kbyte aligned base address of the Universe register space on PCI. Since the Universe register space is only 4 Kbytes, it only responds to this image if address lines [15:12] are zero. PCI address lines [11:0] are used to select the Universe register.

A power-up option determines whether the registers are mapped into Memory or I/O space. There is an internal inversion on the power-up pin (VA[1]). When VA[1] pin is pulled down at power-up, the device starts-up in I/O space and the SPACE bit is set to 1. When VA[1] is pulled up at power-up, the Universe starts-up in Memory space. If mapped into Memory space, the user is free to locate the registers anywhere in the 32-bit address space, but note that the registers are not prefetchable.

Since all Memory and I/O accesses are specified in relation to the Base Address, a write must occur to this register before any Universe registers can be accessed through PCI Memory or I/O space. This initial write can be performed with a PCI configuration transaction or a VMEbus register access.

**Table A.7 : PCI Configuration Miscellaneous 1 Register (PCI\_MISC1)**

Register Name: PCI_MISC1		Offset:03C
Bits	Function	
31-24	MAX_LAT [7:0]	
23-16	MIN_GNT [7:0]	
15-08	INT_PIN [7:0]	
07-00	INT_LINE [7:0]	

**PCI\_MISC1 Description**

Name	Type	Reset By	Reset State	Function
MAX_LAT[7:0]	R	all	0	Maximum Latency: This device has no special latency requirements
MIN_GNT[7:0]	R	all	00000011	Minimum Grant:.250 ns units
INT_PIN[7:0]	R	all	00000001	Interrupt Pin: Universe pin INT# [0] has a PCI compliant I/O buffer
INT_LINE[7:0]	R/W	all	0	Interrupt Line: used by some PCI systems to record interrupt routing information

The MIN\_GNT parameter assumes the Universe master is transferring an aligned burst size of 64 bytes to a 32-bit target with no wait states. This would require roughly 20 clocks (at a clock frequency of 33 MHz, this is about 600 ns). MIN\_GNT is set to three, or 750 ns.

**Table A.8 : PCI Slave Image 0 Control (LSI0\_CTL)**

Register Name: LSI0_CTL				Offset:100
Bits	Function			
31-24	EN	PWEN	Universe Reserved	
23-16	VDW		Universe Reserved	VAS
15-08	PGM		SUPER	Universe Reserved VCT
07-00	Universe Reserved			LAS

**LSI0\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	Power-up Option	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	all	10	VMEbus Maximum Datawidth 00=8 bit data width, 01=16 bit data width, 10=32 bit data width, 11=64 bit data width
VAS	R/W	all	Power-up Option	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	all	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	all	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	all	0	VMEbus Cycle Type 0=Single Cycles Only on VMEbus, 1=Single Cycles and Block Transfers on VMEbus
LAS	R/W	all	Power-up Option	PCI bus Memory Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Type 1 Configuration Space, 11=Reserved

The general, VMEbus and PCI bus controls for this A32 capable image. This PCI slave image has 4 KByte granularity, and dual address cycles are never claimed. Only PCI Memory write transactions can be posted.

**Table A.9 : PCI Slave Image 0 Base Address Register (LSI0\_BS)**

Register Name: LSI0_BS		Offset:104	
Bits	Function		
31-24	BS		
23-16	BS		
15-08	BS	Universe Reserved	
07-00	Universe Reserved		

**LSI0\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:28]	R/W	all	Power-up Option	Base Address
BS[27:12]	R/W	all	0	Base Address

The base address specifies the lowest address in the address range that will be decoded.

**Table A.10 : PCI Slave Image 0 Bound Address Register (LSI0\_BD)**

Register Name: LSI0_BD		Offset:108	
Bits	Function		
31-24	BD		
23-16	BD		
15-08	BD	Universe Reserved	
07-00	Universe Reserved		

**LSI0\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:28]	R/W	all	Power-up Option	Bound Address
BD[27:12]	R/W	all	0	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register. If the bound address is 0, then the addresses decoded are those greater than or equal to the base address.

**Table A.11 : PCI Slave Image 0 Translation Offset (LSI0\_TO)**

Register Name: LSI0_TO		Offset:10C	
Bits	Function		
31-24	TO		
23-16	TO		
15-08	TO	Universe Reserved	
07-00	Universe Reserved		

**LSI0\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	all	0	Translation Offset

The TO is field is used in an unsigned addition with PCIBS and the PCI address to generate a VME address. For example, if PCIBS is 32'h0000\_0000 and TO is 32'h8000\_0000 then the corresponding VME address will be equal to the addition of AD[11:0] and 32'h8000\_0000.

**Table A.12 : PCI Slave Image 1 Control (LSI1\_CTL)**

Register Name: LSI1_CTL				Offset:114
Bits	Function			
31-24	EN	PWEN	Universe Reserved	
23-16	VDW		Universe Reserved	VAS
15-08	PGM		SUPER	Universe Reserved VCT
07-00	Universe Reserved			LAS

**LSI1\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	all	10	VMEbus Maximum Datawidth 00=8 bit data width, 01=16 bit data width, 10=32 bit data width, 11=64 bit data width
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	all	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	all	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	all	0	VMEbus Cycle Type 0=Single Cycles Only on VMEbus, 1=Single Cycles and Block Transfers on VMEbus
LAS	R/W	all	0	PCI bus Memory Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Type 1 Configuration Space, 11=Reserved

The general, VMEbus and PCI bus controls for this A32 capable image. This PCI slave image has 64 KByte granularity (maximum size of 4GBytes), and dual address cycles are never claimed. Only PCI Memory write transactions can be posted.

**Table A.13 : PCI Slave Image 1 Base Address Register (LSI1\_BS)**

Register Name: LSI1_BS		Offset:118
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI1\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address



**Table A.14 : PCI Slave Image 1 Bound Address Register (LSI1\_BD)**

Register Name: LSI1_BD		Offset:11C
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI1\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

**Table A.15 : PCI Slave Image 1 Translation Offset (LSI1\_TO)**

Register Name: LSI1_TO		Offset:120
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI1\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation offset

**Table A.16 : PCI Slave Image 2 Control (LSI2\_CTL)**

Register Name: LSI2_CTL				Offset:128
Bits	Function			
31-24	EN	PWEN	Universe Reserved	
23-16	VDW		Universe Reserved	VAS
15-08	PGM		SUPER	Universe Reserved VCT
07-00	Universe Reserved			LAS

**LSI2\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	all	10	VMEbus Maximum Datawidth 00=8 bit data width, 01=16 bit data width, 10=32 bit data width, 11=64 bit data width
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	all	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	all	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	all	0	VMEbus Cycle Type 0=Single Cycles Only on VMEbus, 1=Single Cycles and Block Transfers on VMEbus
LAS	R/W	all	0	PCI bus Memory Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Type 1 Configuration Space, 11=Reserved

The general, VMEbus and PCI bus controls for this A32 capable image. This PCI slave image has 64 KByte granularity, and dual address cycles are never claimed. Only PCI Memory write transactions can be posted.

**Table A.17 : PCI Slave Image 2 Base Address Register (LSI2\_BS)**

Register Name: LSI2_BS		Offset:12C
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI2\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address

**Table A.18 : PCI Slave Image 2 Bound Address Register (LSI2\_BD)**

Register Name: LSI2_BD		Offset:130
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI2\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

**Table A.19 : PCI Slave Image 2 Translation Offset (LSI2\_TO)**

Register Name: LSI2_TO		Offset:134
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI2\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation offset

**Table A.20 : PCI Slave Image 3 Control (LSI3\_CTL)**

Register Name: LSI3_CTL				Offset:13C
Bits	Function			
31-24	EN	PWEN	Universe Reserved	
23-16	VDW		Universe Reserved	VAS
15-08	PGM		SUPER	Universe Reserved VCT
07-00	Universe Reserved			LAS

**LSI3\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
VDW	R/W	all	10	VMEbus Maximum Datawidth 00=8 bit data width, 01=16 bit data width, 10=32 bit data width, 11=64 bit data width
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=CR/CSR, 110=User1, 111=User2
PGM	R/W	all	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	all	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	all	0	VMEbus Cycle Type 0=Single Cycles Only on VMEbus, 1=Single Cycles and Block Transfers on VMEbus
LAS	R/W	all	0	PCI bus Memory Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Type 1 Configuration Space, 11=Reserved

The general, VMEbus and PCI bus controls for this A32 capable image. This PCI slave image has 64 KByte granularity (maximum size of 4GBytes), and dual address cycles are never claimed. Only PCI Memory write transactions can be posted.

**Table A.21 : PCI Slave Image 3 Base Address Register (LSI3\_BS)**

Register Name: LSI3_BS		Offset:140
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI3\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address



**Table A.22 : PCI Slave Image 3 Bound Address Register (LSI3\_BD)**

Register Name: LSI3_BD		Offset:144
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI3\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

**Table A.23 : PCI Slave Image 3 Translation Offset (LSI3\_TO)**

Register Name: LSI3_TO		Offset:148
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LSI3\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation offset

**Table A.24 : Special Cycle Control Register (SCYC\_CTL)**

<b>Register Name:</b> SCYC_CTL		<b>Offset:</b> 170
<b>Bits</b>	<b>Function</b>	
31-24	Universe Reserved	
23-16	Universe Reserved	
15-08	Universe Reserved	
07-00	Universe Reserved	SCYC

**SCYC\_CTL Description**

<b>Name</b>	<b>Type</b>	<b>Reset By</b>	<b>Reset State</b>	<b>Function</b>
SCYC	R/W	all	0	Special Cycle 00=Disable, 01=RMW, 10=ADOH, 11=Reserved

The special cycle generator will generate an ADOH or RMW cycle for the 32-bit PCI bus address which matches the programmed address in SCYC\_ADDR. A Read-Modify-Write command is initiated by a read to the specified address. Address-Only cycles are initiated by either read or write cycles.

**Table A.25 : Special Cycle PCI bus Address Register**

Register Name: SCYC_ADDR		Offset: 174
Bits	Function	
31-24	ADDR	
23-16	ADDR	
15-08	ADDR	
07-00	ADDR	Universe Reserved

**SCYC\_ADDR Description**

Name	Type	Reset By	Reset State	Function
ADDR[31:2]	R/W	none	x	Address

This register designates the special cycle address. This address must be contained in an enabled PCI image.

**Table A.26 : Special Cycle Swap/Compare Enable Register (SCYC\_EN)**

<b>Register Name:</b> SCYC_EN		<b>Offset:</b> 178
Bits	Function	
31-24	EN	
23-16	EN	
15-08	EN	
07-00	EN	

**SCYC\_EN Description**

Name	Type	Reset By	Reset State	Function
EN[31:0]	R/W	none	x	Bit Enable 0=Disable, 1=Enable

The bits enabled in this register defines the bits that will be involved in the compare and swap operations for VME RMW cycles.

**Table A.27 : Special Cycle Compare Data Register (SCYC\_CMP)**

Register Name: SCYC_CMP		Offset: 17C
Bits	Function	
31-24	CMP	
23-16	CMP	
15-08	CMP	
07-00	CMP	

**SCYC\_CMP Description**

Name	Type	Reset By	Reset State	Function
CMP[31:0]	R/W	none	x	The data returned from the VMEbus is compared with the contents of this register.

The data returned from the read portion of a VMEbus RMW is compared with the contents of this register. SCYC\_EN is used to control which bits are compared.

**Table A.28 : Special Cycle Swap Data Register (SCYC\_SWP)**

Register Name: SCYC_SWP		Offset: 180
Bits	Function	
31-24	SWP	
23-16	SWP	
15-08	SWP	
07-00	SWP	

**SCYC\_SWP Description**

Name	Type	Reset By	Reset State	Function
SWP[31:0]	R/W	none	x	Swap data

If enabled bits matched with the value in the compare register, then the contents of the swap data register is written back to VME. SCYC\_EN is used to control which bits are written back to VME.

**Table A.29 : PCI Miscellaneous Register (LMISC)**

Register Name: LMISC		Offset: 184
Bits	Function	
31-24	CRT	CWT
23-16	Universe Reserved	
15-08	Universe Reserved	
07-00	Universe Reserved	

**LMISC Description**

Name	Type	Reset By	Reset State	Function
CRT	R/W	all	0001	Coupled Request Timer 0000=Disable (Infinite), 0001=128 $\mu$ sec, 0010=256 $\mu$ sec, 0011=512 $\mu$ sec, 0100=1024 $\mu$ sec, 0101=2048 $\mu$ sec, 0110=4096 $\mu$ sec, others=RESERVED
CWT	R/W	all	0	Coupled Window Timer 0000=Disable - release after first coupled transaction, 0001=16 PCI Clocks, 0010=32 PCI Clocks, 0011=64 PCI Clocks, 0100=128 PCI Clocks, 0101=256 PCI Clocks, 0110=512 PCI Clocks, others=RESERVED

If a PCI master generates a coupled transaction and the PCI Slave Channel in the Universe is not currently the owner of the VMEbus, then the Universe retries the PCI master and requests VMEbus ownership. The Universe uses the coupled request timer (CRT) to determine how long to wait after ownership is acquired for a PCI master to retry a coupled transaction. The timer is restarted each time a PCI master attempts the coupled transaction. If this timer expires, then the PCI Slave Channel releases the VME Master Interface.

The Universe uses the coupled window timer (CWT) to determine how long to hold ownership of the VMEbus on behalf of the PCI Slave Channel after processing a coupled transaction. The timer is restarted each time the Universe processes a coupled transaction. If this timer expires, then the PCI Slave Channel releases the VME Master Interface.

Device behaviour is unpredictable if CRT/CWT are changed during coupled cycle activity. These registers should be set only at configuration time, or after disabling all PCI Slave Images.



**Table A.30 : Special PCI Slave Image (SLSI)**

Register Name: SLSI				Offset:188
Bits	Function			
31-24	EN	PWEN	Universe Reserved	
23-16	VDW			Universe Reserved
15-08	PGM			SUPER
07-00	BS			LAS

**SLSI Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
VDW [3:0]	R/W	all	0	VMEbus Maximum Datawidth Each of the four bits specifies a data width for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=16-bit, 1=32-bit
PGM [3:0]	R/W	all	0	Program/Data AM Code Each of the four bits specifies Program/Data AM code for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=Data, 1=Program
SUPER [3:0]	R/W	all	0	Supervisor/User AM Code Each of the four bits specifies Supervisor/User AM code for the corresponding 16 MByte region. Low order bits correspond to the lower address regions. 0=Non-Privileged, 1=Supervisor
BS [5:0]	R/W	all	0	Base Address Specifies a 64 MByte aligned base address for this 64 MByte image.
LAS	R/W	all	0	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Type 1 Configuration Space, 11=Reserved

This register fully specifies an A32 capable special PCI slave image. The base is programmable to a 64 Mbyte alignment, and the size is fixed at 64 Mbytes. Incoming address lines [31:26] (in Memory, I/O, or Configuration space) must match this field for the Universe to decode the access. This special PCI slave image has lower priority than any other PCI slave image.

The 64 Mbytes of the SLSI is partitioned into four 16 Mbyte regions, numbered 0 to 3 (0 is at the lowest address). PCI address bits [25:24] are used to select regions. The top 64 Kbyte of each region is mapped to VMEbus A16 space, and the rest of each 16 Mbyte region is mapped to A24 space.

The user can use the PGM, SUPER and VDW fields to specify the AM code and the maximum port size for each region. The PGM field is ignored for the portion of each region mapped to A16 space.

No block transfer AM codes are generated.

**Table A.31 : PCI Command Error Log Register (L\_CMDERR)**

Register Name: L_CMDERR			Offset: 18C		
Bits	Function				
31-24	CMDERR		M_ERR	Universe Reserved	
23-16	L_STAT	Universe Reserved			
15-08	Universe Reserved				
07-00	Universe Reserved				

**L\_CMDERR Description**

Name	Type	Reset By	Reset State	Function
CMDERR [3:0]	R	all	0111	PCI Command Error Log
M_ERR	R	all	0	Multiple Error Occurred 0=Single error, 1=At least one error has occurred since the logs were frozen.
L_STAT	R/W	all	0	PCI Error Log Status Reads: 0=logs invalid, 1=logs are valid and error logging halted Writes: 0=no effect, 1=clears L_STAT and enables error logging

The Universe PCI Master Interface is responsible for logging errors under the following conditions:

- a posted write transaction results in a target abort,
- a posted write transaction results in a master abort, or
- a maximum retry counter expires during retry of posted write transaction.

This register logs the command information.

**Table A.32 : PCI Address Error Log (LAERR)**

Register Name: LAERR		Offset: 190
Bits	Function	
31-24	LAERR	
23-16	LAERR	
15-08	LAERR	
07-00	LAERR	

**LAERR Description**

Name	Type	Reset By	Reset State	Function
LAERR [31:0]	R	all	0	PCI lower address error log

The starting address of an errored PCI transaction is logged in this register under the following conditions:

- a posted write transaction results in a target abort,
- a posted write transaction results in a master abort, or
- a maximum retry counter expires during retry of posted write transaction.

Contents are qualified by bit L\_STAT of the L\_CMDERR register.

**Table A.33 : DMA Transfer Control Register (DCTL)**

Register Name: DCTL				Offset:200			
Bits	Function						
31-24	L2V	Universe Reserved					
23-16	VDW		Universe Reserved			VAS	
15-08	PGM		SUPER		Universe Reserved		VCT
07-00	LD64EN	Universe Reserved					

**DCTL Description**

Name	Type	Reset By	Reset State	Function
L2V	R/W	all	0	Direction 0=Transfer from VMEbus to PCI Bus, 1=Transfer from PCI bus to VMEbus
VDW	R/W	all	0	VMEbus Maximum Datwidth 00=8 bit data width, 01=16 bit data width, 10=32 bit data width, 11=64 bit data width
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
PGM	R/W	all	0	Program/Data AM Code 00=Data, 01=Program, others=Reserved
SUPER	R/W	all	0	Supervisor/User AM Code 00=Non-Privileged, 01=Supervisor, others=Reserved
VCT	R/W	all	0	VMEbus Cycle Type 0=Single Cycles Only on VMEbus, 1=Single Cycles and Block Transfers on VMEbus
LD64EN	R/W	all	1	Enable 64-bit PCI Bus Transactions 0=Disable, 1=Enable

This register is programmed from either bus or is programmed by the DMAC when it loads the command packet. The DMA only accesses PCI bus Memory space.

**Table A.34 : DMA Transfer Byte Count Register (DTBC)**

Register Name: DTBC		Offset:204
Bits	Function	
31-24	Universe Reserved	
23-16	DTBC	
15-08	DTBC	
07-00	DTBC	

**DTBC Description**

Name	Type	Reset By	Reset State	Function
DTBC[23:0]	R/W	all	0	DMA Transfer Byte Count

This register specifies the number of bytes to be moved by the DMA before the start of the DMA transfer, or the number of remaining bytes in the transfer while the DMA is active. This register is programmed from either bus or is programmed by the DMAC when it loads the command packet. If the register is non-zero at the start of a linked-list operation, these bytes are transferred before reading in the first linked-list entry.

**Table A.35 : DMA PCI Bus Address Register (DLA)**

Register Name: DLA		Offset:208
Bits	Function	
31-24	LA	
23-16	LA	
15-08	LA	
07-00	LA	

**DLA Description**

Name	Type	Reset By	Reset State	Function
LA[31:3]	R/W	none	x	PCI bus Address
LA[2:0]	R/W	all	0	PCI bus Address

This register is programmed from either bus or is programmed by the DMAC when it loads the command packet. During linked-list operation, this register is only updated at the end of a transaction (i.e., processing of a command packet) or when the DMA is stopped or halted.

**Table A.36 : DMA VMEbus Address Register (DVA)**

Register Name: DVA		Offset:210
Bits	Function	
31-24	VA	
23-16	VA	
15-08	VA	
07-00	VA	

**DVA Description**

Name	Type	Reset By	Reset State	Function
VA[31:3]	R/W	none	x	VMEbus Address
VA[2:0]	R/W	all	0	VMEbus Address

This register is programmed from either bus or is programmed by the DMAC when it loads the command packet. During linked-list operation, this register is only updated at the end of a transaction (i.e., processing of a command packet) or when the DMA is stopped or halted.



**Table A.37 : DMA Command Packet Pointer (DCPP)**

Register Name: DCPP		Offset:218
Bits	Function	
31-24	DCPP	
23-16	DCPP	
15-08	DCPP	
07-00	DCPP	Universe Reserved

**DCPP Description**

Name	Type	Reset By	Reset State	Function
DCPP[31:3]	R/W	all	0	DMA Command Packet Pointer

This register contains the pointer into the current command packet. Initially it is programmed to the starting packet of the linked-list, and is updated with the address to a new command packet at the completion of a packet. The packets must be at an aligned 64-bit address.

Table A.38 : DMA General Control/Status Register (DGCS)

Register Name: DGCS					Offset: 220			
Bits	Function							
31-24	GO	STOP_REQ	HALT_REQ	0	CHAIN	0	0	0
23-16	VON				VOFF			
15-08	ACT	STOP	HALT	0	DONE	LERR	VERR	P_ERR
07-00	0	INT_STOP	INT_HALT	0	INT_DONE	INT_LERR	INT_VERR	INT_P_ERR

**DGCS Description**

Name	Type	Reset By	Reset State	Function
GO	W/Read 0 always	all	0	DMA Go Bit 0=No effect, 1=Enable DMA Transfers
STOP_REQ	W/Read 0 always	all	0	DMA Stop Request 0=No effect, 1=Stop DMA transfer when all buffered data has been written
HALT_REQ	W/Read 0 always	all	0	DMA Halt Request 0=No effect, 1=Halt the DMA transfer at the completion of the current command packet
CHAIN	R/W	all	0	DMA Chaining 0=DMA Direct Mode, 1=DMA Linked List mode
VON [3:0]	R/W	all	0	VMEbus Aligned DMA Transfer Count 0000=Until done, 0001=256 bytes, 0010=512 bytes, 0011=1024 bytes, 0100=2048 bytes, 0101=4096 bytes, 0110=8192 bytes, 0111=16384 bytes, others=Reserved
VOFF [3:0]	R/W	all	0	Minimum period the DMA is off the VMEbus between tenures 0000=0μs, 0001=16μs, 0010=32μs, 0011=64μs, 0100=128μs, 0101=256μs, 0110=512μs, 0111=1024μs, others=Reserved
ACT	R	all	0	DMA Active Flag 0=Not Active, 1=Active
STOP	R/Write 1 to Clear	all	0	DMA Stopped Flag 0=Not Stopped, 1=Stopped
HALT	R/Write 1 to Clear	all	0	DMA Halted Flag 0=Not Halted, 1=Halted
DONE	R/Write 1 to Clear	all	0	All DMA Transfers Complete 0=Not Complete, 1=Complete
LERR	R/Write 1 to Clear	all	0	DMA PCI Bus Error 0=No Error, 1=Error
VERR	R/Write 1 to Clear	all	0	DMA VMEbus Error 0=No Error, 1=Error
P_ERR	R/Write 1 to Clear	all	0	Protocol Error Asserted if PCI master interface disabled or lower three bits of PCI and VME addresses differ 0=No Error, 1=Error

**DGCS Description**

<b>Name</b>	<b>Type</b>	<b>Reset By</b>	<b>Reset State</b>	<b>Function</b>
INT_STOP	R/W	all	0	Interrupt when Stopped 0=Disable, 1=Enable
INT_HALT	R/W	all	0	Interrupt when Halted 0=Disable, 1=Enable
INT_DONE	R/W	all	0	Interrupt when Done 0=Disable, 1=Enable
INT_LERR	R/W	all	0	Interrupt on LERR 0=Disable, 1=Enable
INT_VERR	R/W	all	0	Interrupt on VERR 0=Disable, 1=Enable
INT_P_ERR	R/W	all	0	Interrupt on Protocol Error 0=Disable, 1=Enable

STOP, HALT, DONE, LERR, VERR, and P\_ERR all need to be cleared to enable the GO bit to start DMA transfers.

**Table A.39 : DMA Linked List Update Enable Register (D\_LLUE)**

Register Name: D_LLUE		Offset:224
Bits	Function	
31-24	UPDATE	Universe Reserved
23-16	Universe Reserved	
15-08	Universe Reserved	
07-00	Universe Reserved	

**D\_LLUE Description**

Name	Type	Reset By	Reset State	Function
UPDATE	R/W	all	0	DMA Linked List Update Enable 0=PCI Resource not Updating Linked List 1=PCI Resource Updating Linked List

The PCI Resource must read back a logic 1 in the UPDATE field before proceeding to modify the linked list. After the Linked List has been modified the PCI Resource must clear the UPDATE field by writing a logic 0.

**Table A.40 : PCI Interrupt Enable Register (LINT\_EN)**

Register Name: LINT_EN					Offset:300			
Bits	Function							
31-24	Universe Reserved							
23-16								
15-08	ACFAIL	SYSFAIL	SW_INT	SW_JACK	Reserved	VERR	LERR	DMA
07-00	VIRQ7	VIRQ6	VIRQ5	VIRQ4	VIRQ3	VIRQ2	VIRQ1	VOWN

**LINT\_EN Description**

Name	Type	Reset By	Reset State	Function
ACFAIL	R/W	all	0	ACFAIL Interrupt Mask 0=ACFAIL Interrupt masked 1=ACFAIL Interrupt enabled
SYSFAIL	R/W	all	0	SYSFAIL Interrupt Mask 0=SYSFAIL Interrupt masked 1=SYSFAIL Interrupt enabled
SW_INT	R/W	all	0	PCI Software Interrupt Mask 0=PCI Software Interrupt masked 1=PCI Software Interrupt enabled A zero-to-one transition will cause the PCI software interrupt to be asserted. Subsequent zeroing of this bit will cause the interrupt to be masked, but will not clear the PCI Software Interrupt Status bit.
SW_IACK	R/W	all	0	“VME Software IACK” Mask 0 =“VME Software IACK” Interrupt masked 1 =“VME Software IACK” Interrupt enabled
VERR	R/W	all	0	PCI VERR Interrupt Mask 0 =PCI VERR Interrupt masked 1=PCI VERR Interrupt enabled
LERR	R/W	all	0	PCI LERR Interrupt Mask 0 =PCI LERR Interrupt masked 1 =PCI LERR Interrupt enabled
DMA	R/W	all	0	PCI DMA Interrupt Mask 0=PCI DMA Interrupt masked 1=PCI DMA Interrupt enabled
VIRQ7-VIRQ1	R/W	all	0	VIRQx Interrupt Mask 0=VIRQx Interrupt masked 1 =VIRQx Interrupt enabled
VOWN	R/W	all	0	VOWN Interrupt Mask 0=VOWN Interrupt masked 1=VOWN Interrupt Enabled

Bits VIRQ7-VIRQ1 enable the Universe to respond as a VME Interrupt Handler to interrupts on the VIRQ[x] lines. When a VIRQx interrupt is enabled, and the corresponding VIRQ[x] pin is asserted, the Universe requests the VMEbus and performs a VME IACK cycle for that interrupt level. When the interrupt acknowledge cycle completes, the STATUS/ID is stored in the corresponding VINT\_ID register, the VIRQx bit of the LINT\_STAT register is set, and a PCI interrupt is generated. The Universe does not acquire further interrupt STATUS/ID vectors at the same interrupt level until the VIRQx bit in the LINT\_STAT register is cleared.

The other bits enable the respective internal or external sources to interrupt the PCI side.

**Table A.41 : PCI Interrupt Status Register (LINT\_STAT)**

Register Name: LINT_STAT					Offset: 304			
Bits	Function							
31-24	Universe Reserved							
23-16								
15-08	ACFAIL	SYSFAIL	SW_INT	SW_IACK	Reserved	VERR	LERR	DMA
07-00	VIRQ7	VIRQ6	VIRQ5	VIRQ4	VIRQ3	VIRQ2	VIRQ1	VOWN

**LINT\_STAT Description**

Name	Type	Reset By	Reset State	Function
ACFAIL	R	all	0	ACFAIL Interrupt Status/Clear 0: no ACFAIL Interrupt 1: ACFAIL Interrupt active
SYSFAIL	R	all	0	SYSFAIL Interrupt Status/Clear 0: no SYSFAIL Interrupt 1: SYSFAIL Interrupt active
SW_INT	R/Write 1 to Clear	all	0	PCI Software Interrupt Status/Clear 0: no PCI Software Interrupt 1: PCI Software Interrupt active
SW_IACK	R/Write 1 to Clear	all	0	“VME Software IACK” Status/Clear 0: no “VME Software IACK” Interrupt 1: “VME Software IACK” Interrupt active
VERR	R/Write 1 to Clear	all	0	PCI VERR Interrupt Status/Clear 0: no PCI VERR Interrupt 1: PCI VERR Interrupt active
LERR	R/Write 1 to Clear	all	0	PCI LERR Interrupt Status/Clear 0: no PCI LERR Interrupt 1: PCI LERR Interrupt active
DMA	R/Write 1 to Clear	all	0	PCI DMA Interrupt Status/Clear 0: no PCI DMA Interrupt 1: PCI DMA Interrupt active
VIRQ7-VIRQ1	R/Write 1 to Clear	all	0	VIRQ <sub>x</sub> Interrupt Status/Clear 0: no VIRQ <sub>x</sub> Interrupt StatusID vector available 1: VIRQ <sub>x</sub> Interrupt StatusID vector available
VOWN	R/Write 1 to Clear	all	0	VOWN Interrupt Status/Clear 0: no VOWN Interrupt active 1: VOWN Interrupt active

Interrupt status bits indicated as “R” (read only) are level sensitive, and the source of the interrupt must be cleared in order to clear the interrupt status. ACFAIL and SYSFAIL bits directly reflect the status of the respective VMEbus pins (after filtering and synchronization to the PCI clock).

Status bits indicated as “R/Write 1 to Clear” are edge sensitive: the status is latched when the interrupt event occurs. These status bits can be cleared independently of the state of the interrupt source by writing a “1” to the status register. Clearing the status bit does not imply the source of the interrupt is cleared.



Table A.42 : PCI Interrupt Map 0 Register (LINT\_MAP0)

Register Name: LINT_MAP0			Offset: 308	
Bits	Function			
31-24	Reserved	VIRQ7	Reserved	VIRQ6
23-16	Reserved	VIRQ5	Reserved	VIRQ4
15-08	Reserved	VIRQ3	Reserved	VIRQ2
07-00	Reserved	VIRQ1	Reserved	VOWN

LINT\_MAP0 Description

Name	Type	Reset By	Reset State	Function
VIRQ7-VIRQ1	R/W	all	0	PCI interrupt destination (LINT[7:0]) for VIRQx
VOWN	R/W	all	0	VMEbus ownership bit interrupt map to PCI interrupt

The two PCI Interrupt Map registers map various interrupt sources to one of the eight PCI interrupt pins. A value of 000 maps the corresponding interrupt source to LINT# [0], a value of 001 maps to LINT# [1], etc.

**Table A.43 : PCI Interrupt Map 1 Register (LINT\_MAP1)**

Register Name: LINT_MAP1			Offset: 30C		
Bits	Function				
31-24	Reserved	ACFAIL	Reserved	SYSFAIL	
23-16	Reserved	SW_INT	Reserved	SW_IACK	
15-08	Reserved			VERR	
07-00	Reserved	LERR	Reserved	DMA	

**LINT\_MAP1 Description**

Name	Type	Reset By	Reset State	Function
ACFAIL	R/W	all	0	ACFAIL interrupt destination
SYSFAIL	R/W	all	0	SYSFAIL interrupt destination
SW_INT	R/W	all	0	PCI software interrupt destination
SW_IACK	R/W	all	0	VMEbus Software IACK interrupt destination
VERR	R/W	all	0	VMEbus Error interrupt destination
LERR	R/W	all	0	PCI bus Error interrupt destination
DMA	R/W	all	0	DMA interrupt destination

The two PCI Interrupt Map registers map various interrupt sources to one of the eight PCI interrupt pins. A value of 000 maps the corresponding interrupt source to LINT# [0], a value of 001 maps to LINT# [1], etc.

**Table A.44 : VMEbus Interrupt Enable Register (VINT\_EN)**

Register Name: VINT_EN					Offset:310				
Bits	Function								
31-24	Universe Reserved								
23-16									
15-08	Reserved			SW_INT	Reserved	VERR	LERR	DMA	
07-00	LINT7	LINT6	LINT5	LINT4	LINT3	LINT2	LINT1	LINT0	

**VINT\_EN Description**

Name	Type	Reset By	Reset State	Function
SW_INT	R/W	all	Power-up Option	“VME Software Interrupt” Mask 0 = VME Software Interrupt masked 1 = VME Software Interrupt enabled A zero-to-one transition causes the VME software interrupt to be asserted. Subsequent zeroing of this bit causes the interrupt to be masked and the VMEbus interrupt negated, but does not clear the VME software interrupt status bit.
VERR	R/W	all	0	VERR Interrupt Mask 0 =PCI VERR Interrupt masked 1=PCI VERR Interrupt enabled
LERR	R/W	all	0	LERR Interrupt Mask 0 =PCI LERR Interrupt masked 1 =PCI LERR Interrupt enabled
DMA	R/W	all	0	DMA Interrupt Mask 0=PCI DMA Interrupt masked 1=PCI DMA Interrupt enabled
LINT7-LINT0	R/W	all	0	PCI Interrupt Mask 0=LINTx Interrupt masked 1 =LINTx Interrupt enabled

This register enables the various sources of VMEbus interrupts. SW\_INT can be enabled with the VME64AUTO power-up option.

**Table A.45 : VMEbus Interrupt Status Register (VINT\_STAT)**

Register Name: VINT_STAT					Offset:314				
Bits	Function								
31-24	Universe Reserved								
23-16									
15-08	Reserved			SW_INT	Reserved	VERR	LERR	DMA	
07-00	LINT7	LINT6	LINT5	LINT4	LINT3	LINT2	LINT1	LINT0	

**VINT\_STAT Description**

Name	Type	Reset By	Reset State	Function
SW_INT	R/Write 1 to Clear	all	Power-up Option	VME Software Interrupt Status/Clear 0=no VME Software Interrupt 1=VME Software Interrupt active
VERR	R/Write 1 to Clear	all	0	VERR Interrupt Status/Clear 0=noVME VERR Interrupt 1=VME VERR Interrupt active
LERR	R/Write 1 to Clear	all	0	LERR Interrupt Status/Clear 0=noVME LERR Interrupt 1=VME LERR Interrupt active
DMA	R/Write 1 to Clear	all	0	DMA Interrupt Status/Clear 0=noVME DMA Interrupt 1=VME DMA Interrupt active
LINT7-LINT0	R/Write 1 to Clear	all	0	LINT <sub>x</sub> Interrupt Status/Clear 0=no LINT <sub>x</sub> Interrupt 1=LINT <sub>x</sub> Interrupt active

SW\_INT can be set with the VME64AUTO power-up option.

Table A.46 : VME Interrupt Map 0 Register (VINT\_MAP0)

Register Name: VINT_MAP0			Offset: 318	
Bits	Function			
31-24	Reserved	LINT7	Reserved	LINT6
23-16	Reserved	LINT5	Reserved	LINT4
15-08	Reserved	LINT3	Reserved	LINT2
07-00	Reserved	LINT1	Reserved	LINT0

VINT\_MAP0 Description

Name	Type	Reset By	Reset State	Function
LINT7-LINT0	R/W	all	0	VMEbus destination of PCI bus interrupt source

The two VME Interrupt Map registers map various interrupt sources to one of the seven VMEbus interrupt pins. A value of 001 maps the corresponding interrupt source to VIRQ\*[1], a value of 002 maps to VIRQ\*[2], etc. A value of 000 effectively masks the interrupt since there is no corresponding VIRQ\*[0].

**Table A.47 : VME Interrupt Map 1 Register (VINT\_MAP1)**

Register Name: VINT_MAP1				Offset: 31C
Bits	Function			
31-24	Reserved			
23-16	Reserved			SW_INT
15-08	Reserved			VERR
07-00	Reserved	LERR	Reserved	DMA

**VINT\_MAP1 Description**

Name	Type	Reset By	Reset State	Function
SW_INT	R/W	all	Power-up Option	VMEbus Software interrupt destination
VERR	R/W	all	0	VMEbus Error interrupt destination
LERR	R/W	all	0	PCI bus Error interrupt destination
DMA	R/W	all	0	DMA interrupt destination

The two VME Interrupt Map registers map various interrupt sources to one of the seven VMEbus interrupt pins. A value of 001 maps the corresponding interrupt source to VIRQ\*[1], a value of 002 maps to VIRQ\*[2], etc. A value of 000 effectively masks the interrupt since there is no corresponding VIRQ\*[0].

SW\_INT is set to 010 with the VME64AUTO power-up option.

Table A.48 : Interrupt STATUS/ID Out (STATID)

Register Name: STATID		Offset: 320
Bits	Function	
31-24	STATID [7:1]	Reserved
23-16	Universe Reserved	
15-08	Universe Reserved	
07-00	Universe Reserved	

STATID Description

Name	Type	Reset By	Reset State	Function
STATID [7:1]	R/W	all	1111111	Bits [7:1] of the STATUS/ID byte are returned when the Universe responds to a VMEbus IACK cycle.

When the Universe responds to an interrupt acknowledge cycle on VMEbus it returns an 8 bit STATUS/ID. The upper seven bits are specified by STATID field and can be written by software to uniquely identify the VMEbus module within the system. The low order bit of the STATUS/ID is cleared if the Universe is generating a software interrupt (SW\_IACK) at the same level as the interrupt acknowledge cycle, otherwise it is set.

The reset state is designed to support the VME64 Auto ID STATUS/ID value.

**Table A.49 : VIRQ1 STATUS/ID Register (V1\_STATID)**

Register Name: V1_STATID		Offset: 324
Bits	Function	
31-24	Universe Reserved	
23-16	Universe Reserved	
15-08	Universe Reserved	ERR
07-00	STATID [7:0]	

**V1\_STATID Description**

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 1 VMEbus interrupt

The V<sub>x</sub>\_STATID registers are read-only registers that hold the 8-bit VMEbus STATUS/ID that is acquired when the Universe performs a IACK cycle for a given interrupt level. The Universe is enabled as the interrupt handler for a given interrupt level via the VIRQ<sub>x</sub> bits of the LINT\_EN register. Once a vector for a given level is acquired, the Universe does not perform a subsequent interrupt acknowledge cycle at that level until the corresponding VIRQ<sub>x</sub> bit in the LINT\_STAT register is cleared.

The acquisition of a level *x* STATUS/ID by the Universe updates the STATUS/ID field of the corresponding V<sub>x</sub>\_STATID register and generation of a PCI interrupt. A VMEbus error during the acquisition of the STATUS/ID vector sets the ERR bit, which means the STATUS/ID field may not contain a valid vector.



Table A.50 : VIRQ2 STATUS/ID Register (V2\_STATID)

Register Name: V2_STATID		Offset: 328	
Bits	Function		
31-24	Universe Reserved		
23-16	Universe Reserved		
15-08	Universe Reserved		ERR
07-00	STATID [7:0]		

V2\_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 2 VMEbus interrupt

**Table A.51 : VIRQ3 STATUS/ID Register (V3\_STATID)**

Register Name: V3_STATID		Offset: 32C	
Bits	Function		
31-24	Universe Reserved		
23-16	Universe Reserved		
15-08	Universe Reserved		ERR
07-00	STATID [7:0]		

**V3\_STATID Description**

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 3VMEbus interrupt

Table A.52 : VIRQ4 STATUS/ID Register (V4\_STATID)

Register Name: V4_STATID		Offset: 330	
Bits	Function		
31-24	Universe Reserved		
23-16	Universe Reserved		
15-08	Universe Reserved		ERR
07-00	STATID [7:0]		

V4\_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 4 VMEbus interrupt

**Table A.53 : VIRQ5 STATUS/ID Register (V5\_STATID)**

Register Name: V5_STATID		Offset: 334
Bits	Function	
31-24	Universe Reserved	
23-16	Universe Reserved	
15-08	Universe Reserved	ERR
07-00	STATID [7:0]	

**V5\_STATID Description**

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 5 VMEbus interrupt

Table A.54 : VIRQ6 STATUS/ID Register (V6\_STATID)

Register Name: V6_STATID		Offset: 338	
Bits	Function		
31-24	Universe Reserved		
23-16	Universe Reserved		
15-08	Universe Reserved		ERR
07-00	STATID [7:0]		

V6\_STATID Description

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 6 VMEbus interrupt

**Table A.55 : VIRQ7 STATUS/ID Register (V7\_STATID)**

Register Name: V7_STATID		Offset: 33C	
Bits	Function		
31-24	Universe Reserved		
23-16	Universe Reserved		
15-08	Universe Reserved		ERR
07-00	STATID [7:0]		

**V7\_STATID Description**

Name	Type	Reset By	Reset State	Function
ERR	R	all	0	Error Status Bit 0=STATUS/ID was acquired without bus error 1=bus error occurred during acquisition of the STATUS/ID
STATID [7:0]	R	all	0	STATUS/ID acquired during IACK cycle for level 7 VMEbus interrupt

**Table A.56 : Master Control Register (MAST\_CTL)**

Register Name: MAST_CTL				Offset: 400		
Bits	Function					
31-24	MAXRTRY			PWON		
23-16	VRL	VRM	VREL	VOWN	VOWN_ACK	Universe Reserved
15-08	Universe Reserved		PABS	Universe Reserved		
07-00	BUS_NO					

**MAST\_CTL Description**

Name	Type	Reset By	Reset State	Function
MAXRTRY [3:0]	R/W	all	1000	Maximum Number of Retries 0000=Retry Forever, Multiples of 64 (0001 through 1111). Maximum Number of retries before the PCI master interface signals error condition
PWON [3:0]	R/W	all	0000	Posted Write Transfer Count 0000=128 bytes, 0001=256 bytes, 0010=512 bytes, 0011=1024 bytes, 0100=2048 bytes, 0101=4096 bytes, Others=Reserved Transfer count at which the PCI Slave Channel Posted Writes FIFO gives up the VME Master Interface.
VRL [1:0]	R/W	all	11	VMEbus Request Level 00=Level 0, 01=Level 1, 10=Level 2, 11=Level 3
VRM	R/W	all	0	VMEbus Request Mode 0=Demand, 1=Fair
VREL	R/W	all	0	VMEbus Release Mode 0=Release When Done, 1=Release on Request
VOWN	W	all	0	VME Ownership Bit 0=Release VMEbus, 1=Acquire and Hold VMEbus
VOWN_ACK	R	all	0	VME Ownership Bit Acknowledge 0=VMEbus not owned, 1=VMEbus acquired and held due to assertion of VOWN
PABS	R/W	all	0	PCI Aligned Burst Size 0=32-byte, 1=64-byte Controls the PCI address boundary at which the Universe breaks up a PCI transaction. See "Posted Writes" on page 2-38. and "FIFO Operation and Bus Ownership" on page 2-82.
BUS_NO [7:0]	R/W	all	0000 0000	PCI Bus Number

VOWN\_ACK is synchronized to the PCI clock.

VMEbus masters should not set the VOWN bit since this will lock up the VMEbus.

BUS\_NO is used by the PCI master interface when it is mapping VME transactions into PCI Configuration space. If the bus number of the PCI address (bits [23:16]) is equal to the BUS\_NO field, then the Universe generates a Type 0 configuration cycle, otherwise Type 1 is generated.



**Table A.57 : Miscellaneous Control Register (MISC\_CTL)**

Register Name: MISC_CTL					Offset: 404			
Bits	Function							
31-24	VBTO				Reserved	VARB	VARBTO	
23-16	SW_LRST	SW_SRST	Reserved	BI	ENGBI	RESCIND	SYSCON	V64AUTO
15-08	Universe Reserved							
07-00	Universe Reserved							

**MISC\_CTL Description**

Name	Type	Reset By	Reset State	Function
VBTO	R/W	all	0011	VME Bus Time-out 0000=Disable, 0001=16 $\mu$ sec, 0010=32 $\mu$ sec, 0011=64 $\mu$ sec, 0100=128 $\mu$ sec, 0101=256 $\mu$ sec, 0110=512 $\mu$ sec, 0111=1024 $\mu$ sec, others=RESERVED
VARB	R/W	all	0	VMEbus Arbitration Mode 0=Round Robin, 1=Priority
VARBTO	R/W	all	01	VMEbus Arbitration Time-out 00=Disable Timer, 01=16 $\mu$ s, 10=256 $\mu$ s, others=Reserved
SW_LRST	W	all	0	Software PCI Reset 0=No affect, 1=Initiate LRST# A read always returns 0.
SW_SYSRST	W	all	0	Software VMEbus SYSRESET 0=No affect, 1=Initiate SYSRST* A read always returns 0.
BI	R/W	all	Power-up Option	BI-Mode 0=Universe is not in BI-Mode, 1=Universe is in BI-Mode Write to this bit to change the Universe BI-Mode status. This bit is also affected by the global BI-Mode initiator VRIRQ1*, if this feature is enabled.
ENGBI	R/W	all	0	Enable Global BI-Mode Initiator 0=Assertion of VIRQ1 ignored, 1=Assertion of VIRQ1 puts device in BI-Mode
RESCIND	R/W	all	1	Rescinding DTACK Enable 0=Disable, 1=Enable
SYSCON	R/W	all	Power-up Option	SYSCON 0=Universe is not VMEbus System Controller, 1=Universe is VMEbus System Controller
V64AUTO	R/W	all	Power-up Option	VME64 Auto ID Write: 0=No effect, 1=Initiate sequence This bit initiates Universe VME64 Auto ID Slave participation.

VMEbus masters should not write to SW\_SYSRST, and PCI masters should not write to SW\_LRST.

The bits VBTO, VARB and VARBTO support SYSCON functionality.

Universe participation in the VME64 Auto ID mechanism is controlled by the VME64AUTO bit. When this bit is detected high, the Universe uses the SW\_IACK mechanism to generate VXIRQ2 on the VMEbus, then releases VXSYSFAIL. Access to the CR/CSR image is enabled when the level 2 interrupt acknowledge cycle completes. This sequence can be initiated with a power-up option or by software writing a 1 to this bit.

**Table A.58 : Miscellaneous Status Register (MISC\_STAT)**

Register Name: MISC_STAT					Offset: 408				
Bits	Function								
31-24	ENDIAN	LCLSIZE	Reserved		DY4AUTO	Universe Reserved			
23-16	Universe Reserved		MYBBSY	Reserved	DY4_DONE	TXFE	RXFE	Reserved	
15-08	DY4AUTOID								
07-00	Universe Reserved								

**MISC\_STAT Description**

Name	Type	Reset By	Reset State	Function
ENDIAN	R	all	0	The Universe is always little endian on the PCI bus
LCLSIZE	R	all	Power-up Option	PCI Bus Size At the trailing edge of RST#, the Universe samples REQ64# to determine the PCI bus size. This bit reflects the result. 0=32-bits, 1=64-bits
DY4AUTO	R	all	Power-up Option	DY4 Auto ID Enable 0=Disable, 1=Enable
MYBBSY	R	all	1	Universe BBSY 0=Asserted, 1=Negated
DY4DONE	R	all	0	DY4 Auto ID Done 0=Not done, 1=Done
TXFE	R	all	1	Transmit FIFO Empty 0=Not empty, 1=Empty
RXFE	R	all	1	Receive FIFO Empty 0=Not Empty, 1=Empty
DY4AUTOID	R	none	x	DY4 Auto ID

**Table A.59 : User AM Codes Register (USER\_AM)**

Register Name: USER_AM		Offset: 40C
Bits	Function	
31-24	USER1AM	Universe Reserved
23-16	USER2AM	Universe Reserved
15-08	Universe Reserved	
07-00	Universe Reserved	

**USER\_AM Description**

Name	Type	Reset By	Reset State	Function
USER1AM [5:0]	R/W	all	010000	User AM Code 1
USER2AM [5:0]	R/W	all	010000	User AM Code 2

The reset state is one of the VME64 user-defined AM codes.

**Table A.60 : VMEbus Slave Image 0 Control (VSI0\_CTL)**

Register Name: VSI0_CTL				Offset: F00	
Bits	Function				
31-24	EN	PWEN	PREN	Universe Reserved	
23-16	PGM		SUPER	Reserved	VAS
15-08	Universe Reserved				
07-00	LD64EN	LLRMW	Universe Reserved		LAS

**VSI0\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	all	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	all	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	all	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	all	0	Enable 64-bit PCI bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	all	0	Enable PCI bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	all	0	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only transactions destined for PCI Memory space are decoupled (the posted write RXFIFO generates on Memory space transactions on the PCI bus). This image has 4 Kbyte resolution.

In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).

**Table A.61 : VMEbus Slave Image 0 Base Address Register (VSI0\_BS)**

Register Name: VSI0_BS		Offset: F04	
Bits	Function		
31-24	BS		
23-16	BS		
15-08	BS	Universe Reserved	
07-00	Universe Reserved		

**VSI0\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	none	x	Base Address

The base address specifies the lowest address in the address range that will be decoded.

**Table A.62 : VMEbus Slave Image 0 Bound Address Register (VSI0\_BD)**

Register Name: VSI0_BD		Offset: F08	
Bits	Function		
31-24	BD		
23-16	BD		
15-08	BD	Universe Reserved	
07-00	Universe Reserved		

**VSI0\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:12]	R/W	none	x	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

**Table A.63 : VMEbus Slave Image 0 Translation Offset (VSI0\_TO)**

Register Name: VSI0_TO		Offset: F0C	
Bits	Function		
31-24	TO		
23-16	TO		
15-08	TO	Universe Reserved	
07-00	Universe Reserved		

**VSI0\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:12]	R/W	none	x	Translation Offset



**Table A.64 : VMEbus Slave Image 1 Control (VSI1\_CTL)**

Register Name: VSI1_CTL				Offset: F14
Bits	Function			
31-24	EN	PWEN	PREN	Universe Reserved
23-16	PGM		SUPER	Reserved VAS
15-08	Universe Reserved			
07-00	LD64EN	LLRMW	Universe Reserved LAS	

**VSI1\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	all	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	all	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	all	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	all	1	Enable 64-bit PCI bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	all	1	Enable PCI bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	all	0	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only PCI Memory space accesses are decoupled (the posted write RXFIFO generates PCI transactions only to Memory space). This image has a 64 Kbyte resolution (maximum size of image is 4 GBytes).

In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).

**Table A.65 : VMEbus Slave Image 1 Base Address Register (VSI1\_BS)**

Register Name: VSI1_BS		Offset: F18
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI1\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address

The base address specifies the lowest address in the address range that will be decoded.

**Table A.66 : VMEbus Slave Image 1 Bound Address Register (VSI1\_BD)**

Register Name: VSI1_BD		Offset: F1C
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI1\_BD Description**

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

The addresses decoded in a slave image are those which are greater than or equal to the base address and less than the bound register.

**Table A.67 : VMEbus Slave Image 1 Translation Offset (VSI1\_TO)**

Register Name: VSI1_TO		Offset: F20
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI1\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation Offset

**Table A.68 : VMEbus Slave Image 2 Control (VSI2\_CTL)**

Register Name: VSI2_CTL				Offset: F28
Bits	Function			
31-24	EN	PWEN	PREN	Universe Reserved
23-16	PGM		SUPER	Reserved VAS
15-08	Universe Reserved			
07-00	LD64EN	LLRMW	Universe Reserved LAS	

**VSI2\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	all	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	all	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	all	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	all	0	Enable 64-bit PCI bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	all	0	Enable PCI bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	all	0	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only PCI Memory space accesses are decoupled (the posted write RXFIFO generates PCI transactions only to Memory space). This image has a 64 Kbyte resolution (maximum size of image is 4 GBytes).

In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).

**Table A.69 : VMEbus Slave Image 2 Base Address Register (VSI2\_BS)**

Register Name: VSI2_BS		Offset: F2C
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI2\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address

Table A.70 : VMEbus Slave Image 2 Bound Address Register (VSI2\_BD)

Register Name: VSI2_BD		Offset: F30
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

VSI2\_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

**Table A.71 : VMEbus Slave Image 2 Translation Offset (VSI2\_TO)**

Register Name: VSI2_TO		Offset: F34
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI2\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation Offset



**Table A.72 : VMEbus Slave Image 3 Control (VSI3\_CTL)**

Register Name: VSI3_CTL				Offset: F3C	
Bits	Function				
31-24	EN	PWEN	PREN	Universe Reserved	
23-16	PGM		SUPER	Reserved	VAS
15-08	Universe Reserved				
07-00	LD64EN	LLRMW	Universe Reserved		LAS

**VSI3\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	all	0	Image Enable 0=Disable, 1=Enable
PWEN	R/W	all	0	Posted Write Enable 0=Disable, 1=Enable
PREN	R/W	all	0	Prefetch Read Enable 0=Disable, 1=Enable
PGM	R/W	all	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	all	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	all	0	VMEbus Address Space 000=A16, 001=A24, 010=A32, 011= Reserved, 100=Reserved, 101=Reserved, 110=User1, 111=User2
LD64EN	R/W	all	0	Enable 64-bit PCI bus Transactions 0=Disable, 1=Enable
LLRMW	R/W	all	0	Enable PCI bus Lock of VMEbus RMW 0=Disable, 1=Enable
LAS	R/W	all	0	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Configuration Space, 11=Reserved

This register provides the general, VMEbus and PCI controls for this slave image. Note that only PCI Memory space accesses are decoupled (the posted write RXFIFO generates PCI transactions only to Memory space). This image has a 64 Kbyte resolution (maximum size of image is 4 GBytes).

In order for a VMEbus slave image to respond to an incoming cycle, the PCI master interface must be enabled (bit BM in the PCI\_CSR register, Table A.3).

**Table A.73 : VMEbus Slave Image 3 Base Address Register (VSI3\_BS)**

Register Name: VSI3_BS		Offset: F40
Bits	Function	
31-24	BS	
23-16	BS	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI3\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:16]	R/W	none	x	Base Address

Table A.74 : VMEbus Slave Image 3 Bound Address Register (VSI3\_BD)

Register Name: VSI3_BD		Offset: F44
Bits	Function	
31-24	BD	
23-16	BD	
15-08	Universe Reserved	
07-00	Universe Reserved	

VSI3\_BD Description

Name	Type	Reset By	Reset State	Function
BD[31:16]	R/W	none	x	Bound Address

**Table A.75 : VMEbus Slave Image 3 Translation Offset (VSI3\_TO)**

Register Name: VSI3_TO		Offset: F48
Bits	Function	
31-24	TO	
23-16	TO	
15-08	Universe Reserved	
07-00	Universe Reserved	

**VSI3\_TO Description**

Name	Type	Reset By	Reset State	Function
TO[31:16]	R/W	none	x	Translation Offset

Table A.76 : VMEbus Register Access Image Control Register (VRAI\_CTL)

Register Name: VRAI_CTL			Offset: F70		
Bits	Function				
31-24	EN	Universe Reserved			
23-16	PGM		SUPER	Universe Reserved	VAS
15-08	Universe Reserved				
07-00	Universe Reserved				

VRAI\_CTL Description

Name	Type	Reset By	Reset State	Function
EN	R/W	all	Power-up Option	Image Enable 0=Disable, 1=Enable
PGM	R/W	all	11	Program/Data AM Code 00=Reserved, 01=Data, 10=Program, 11=Both
SUPER	R/W	all	11	Supervisor/User AM Code 00=Reserved, 01=Non-Privileged, 10=Supervisor, 11=Both
VAS	R/W	all	Power-up Option	VMEbus Address Space 00=A16, 01=A24, 10=A32, all others are reserved

The VME Register Access Image allows access to the Universe registers with standard VMEbus cycles. Only single cycle and lock AM codes are accepted. When a register is accessed with a RMW, it is locked for the duration of the transaction.

**Table A.77 : VMEbus Register Access Image Base Address Register (VRAI\_BS)**

Register Name: VRAI_BS		Offset: F74	
Bits	Function		
31-24	BS		
23-16	BS		
15-08	BS	Universe Reserved	
07-00	Universe Reserved		

**VRAI\_BS Description**

Name	Type	Reset By	Reset State	Function
BS[31:12]	R/W	all	Power-up Option	The base address specifies the lowest address in the 4 Kbyte VMEbus Register Access Image.

The base address specifies the lowest address in the 4 Kbyte VMEbus Register Access Image. The reset state is a function of the Power-up Option behaviour of the VAS field in VRAI\_CTL:

VRAI_CTL: VAS	BS [31:24]	BS [23:16]	BS [15:12]
A16	0	0	Power-up Option VA [28:25]
A24	0	Power-up Option VA [28:21]	0
A32	Power-up Option VA [28:21]	0	0

**Table A.78 : VMEbus CSR Control Register (VCSR\_CTL)**

Register Name: VCSR_CTL		Offset: F80
Bits	Function	
31-24	EN	Universe Reserved
23-16	Universe Reserved	
15-08	Universe Reserved	
07-00	Universe Reserved	LAS

**VCSR\_CTL Description**

Name	Type	Reset By	Reset State	Function
EN	R/W	PWR,VME	0	Image Enable 0=Disable, 1=Enable
LAS	R/W	all	Power-up Option	PCI bus Address Space 00=PCI bus Memory Space, 01=PCI bus I/O Space, 10=PCI bus Configuration Space, 11=Reserved

The EN bit allows software to enable or disable the Universe CR/CSR image. This image can also be enabled by the VME64 Auto ID process.

For CSR's not supported in the Universe and for CR accesses, the LAS field determines the PCI bus command that will be generated when the cycle is mapped to the PCI bus.

**Table A.79 : VMEbus CSR Translation Offset (VCSR\_TO)**

Register Name: VCSR_TO		Offset: F84	
Bits	Function		
31-24	TO		
23-16	TO	Universe Reserved	
15-08	Universe Reserved		
07-00	Universe Reserved		

**VCSR\_TO Description**

Name	Type	Reset By	Reset State	Function
TO [31:24]	R/W	PWR, VME	0	Translation Offset
TO [23:19]	R/W	all	Power-up Option	Translation Offset

For CSR's not supported in the Universe and for CR accesses, the translation offset is added to the 24-bit VMEbus address to produce a 32-bit PCI bus address. Negative offsets are not supported.



Table A.80 : VMEbus AM Code Error Log (V\_AMERR)

Register Name: V_AMERR			Offset: F88		
Bits	Function				
31-24	AMERR			IACK	M_ERR
23-16	V_STAT	Universe Reserved			
15-08	Universe Reserved				
07-00	Universe Reserved				

V\_AMERR Description

Name	Type	Reset By	Reset State	Function
AMERR [5:0]	R	all	0	VMEbus AM Code Error Log
IACK	R	all	0	VMEbus IACK Signal
M_ERR	R	all	0	Multiple Error Occurred 0=Single error, 1=At least one error has occurred since the logs were frozen
V_STAT	R/W	all	0	VME Error Log Status Reads: 0=logs invalid, 1=logs are valid and error logging halted Writes: 0=no effect, 1=clears V_STAT and enables error logging

The Universe VMEbus Master Interface is responsible for logging the parameters of a posted write transaction that results in a bus error. This register holds the address modifier code and the state of the IACK\* signal. The register contents are qualified by the V\_STAT bit.

**Table A.81 : VMEbus Address Error Log (VAERR)**

Register Name: VAERR		Offset: F8C
Bits	Function	
31-24	VAERR	
23-16	VAERR	
15-08	VAERR	
07-00	VAERR	

**VAERR Description**

Name	Type	Reset By	Reset State	Function
VAERR [31:0]	R	all	0	VMEbus address error log

The Universe PCI Master Interface is responsible for logging the parameters of a posted write transaction that results in a bus error. This register holds the address. The register contents are qualified by the V\_STAT bit of the V\_AMERR register.

**Table A.82 : VMEbus CSR Bit Clear Register (VCSR\_CLR)**

Register Name: VCSR_CLR				Offset: FF4
Bits	Function			
31-24	RESET	SYSFAIL	FAIL	Universe Reserved
23-16	Universe Reserved			
15-08	Universe Reserved			
07-00	Universe Reserved			

**VCSR\_CLR Description**

Name	Type	Reset By	Reset State	Function
RESET	R/W	PWR, VME	0	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=negate LRST#
SYSFAIL	R/W	all	Power-up Option	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes: 0=no effect, 1=negate VXSYSFAIL
FAIL	R	PWR, VME	0	Board Fail 0=Board has not failed

This register implements the Bit Clear Register as defined in the VME64 specification. Note that the RESET bit can be written to only from the VMEbus.

**Table A.83 : VMEbus CSR Bit Set Register (VCSR\_SET)**

Register Name: VCSR_SET				Offset: FF8
Bits	Function			
31-24	RESET	SYSFAIL	FAIL	Universe Reserved
23-16	Universe Reserved			
15-08	Universe Reserved			
07-00	Universe Reserved			

**VCSR\_SET Description**

Name	Type	Reset By	Reset State	Function
RESET	R/W	PWR, VME	0	Board Reset Reads: 0=LRST# not asserted, 1=LRST# asserted Writes: 0=no effect, 1=assert LRST#
SYSFAIL	R/W	all	Power-up Option	VMEbus SYSFAIL Reads: 0=VXSYSFAIL not asserted, 1=VXSYSFAIL asserted Writes: 0=no effect, 1=assert VXSYSFAIL
FAIL	R	PWR, VME	0	Board Fail 0=Board has not failed

This register implements the Bit Set Register as defined in the VME64 specification. Note that the RESET bit can be written to only from the VMEbus. Writing 1 to the RESET bit asserts LRST#. The PCI reset remains asserted until a 1 is written to the RESET bit of the VCSR\_CLR register.

Table A.84 : VMEbus CSR Base Address Register (VCSR\_BS)

Register Name: VCSR_BS		Offset: FFC	
Bits	Function		
31-24	BS	Universe Reserved	
23-16	Universe Reserved		
15-08	Universe Reserved		
07-00	Universe Reserved		

VCSR\_BS Description

Name	Type	Reset By	Reset State	Function
BS [23:19]	R/W	PWR, VME	0	Base Address

The base address specifies one of thirty-one available CR/CSR windows as defined in the VME64 specification. Each window consumes 512 Kbytes of CR/CSR space.



## Appendix B Performance

As a VMEbus bridge, the Universe's most important function is data transfer. This function is performed by its three channels: the PCI Slave Channel, the VME Slave Channel, and the DMA Channel. Since each channel operates independently of the others and because each has its own unique characteristics, the following analysis reviews the data transfer performance for each channel. Where relevant, descriptions of factors affecting performance and how they might be controlled in different environments are discussed.

The decoupled nature of the Universe can cause some confusion in discussing performance parameters. This is because, in a fully decoupled bus bridge each of the two opposing buses operates at its peak performance independently of the other. The Universe, however, because of the finite size of its FIFOs does not represent a 100% decoupled bridge. As the FIFOs fill or empty (depending on the direction of data movement) the two buses tend to migrate to matched performance where the higher performing bus is forced to slow down to match the other bus. This limits the sustained performance of the device. Some factors such as the PCI Aligned Burst Size and VME request/release modes can limit the effect of FIFO size and enhance performance.

Another aspect in considering the performance of a device is bandwidth consumption. The greater bandwidth consumed to transfer a given amount of data, the less is available for other bus masters. Decoupling significantly improves the Universe's bandwidth consumption, and on the PCI bus allows it to use the minimum permitted by the PCI specification.

To simplify the analysis and allow comparison with other devices, Universe performance has been calculated using the following assumptions:

As a PCI master:

- one clock bus grant latency
- zero wait state PCI target

As a VME master:

- ideal VME slave response ( $DS^*$  to  $DTACK^* = 30ns$ )

Assumed as part of any calculation on VME performance is the inclusion of VME transceivers with propagation delay of 4 ns.

## B.1 PCI Slave Channel

### B.1.1 Coupled Cycles

#### B.1.1.1 Request of VMEbus

The Universe has a "Coupled Window Timer" (CWT in the LMISC register) which permits the coupled channel to maintain ownership of the VMEbus for an extended period beyond the completion of a cycle. This permits subsequent coupled accesses to the VMEbus to occur back-to-back without requirement for re-arbitration.

The CWT should be set for the expected latency between sequential coupled accesses attempted by the CPU. In calculating the latency expected here, the designer needs to account for latency across their host PCI bridge as well as latency encountered in re-arbitration for the PCI bus between each coupled access. Care must be taken not to set the CWT greater than necessary as the Universe blocks all decoupled write transactions with target-retry, while the coupled channel owns the VMEbus. It is only when the CWT has expired that the PCI bus is permitted to enqueue transactions in the TXFIFO.

When a coupled access to the VMEbus is attempted, the Universe generates a target-retry to the PCI initiator if the coupled path does not currently own the VMEbus. This occurs if the Universe is not currently VMEbus master, or if the DMA is currently VMEbus master or if entries exist in the TXFIFO.

If the Universe does not have ownership of the VMEbus when a coupled access is attempted, the Universe generates a target-retry with a single wait state (See Figure B.1). The request for the VMEbus occurs six CLK64 periods after the cycle is retried.

#### B.1.1.2 Read Cycles

Once the coupled channel owns the VMEbus, the Universe propagates the cycle out to the VMEbus. Figure B.1 shows such a coupled read cycle against an ideal VME slave. There are 24 wait states inserted by the Universe on the PCI bus before it responds with TRDY#. Further wait states are inserted for each extra 30ns in slave response.

Performing 32-bit PCI reads from VME gives a sustained performance of approximately 4 MB/s. Figure B.2 shows several of these access occurring consecutively.



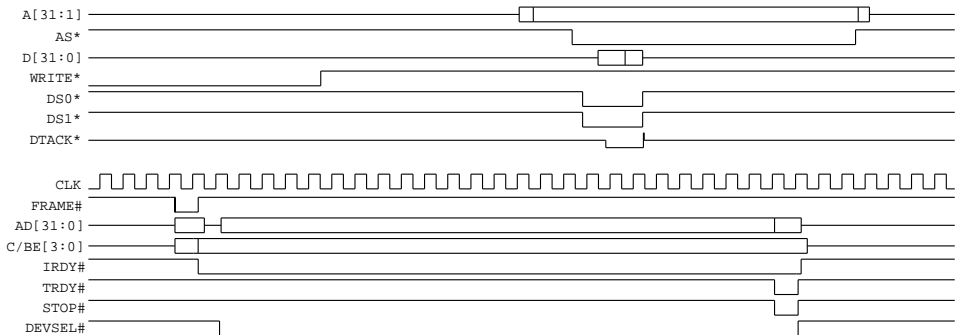


Figure B.1 : Coupled Read Cycle - Universe as VME Master

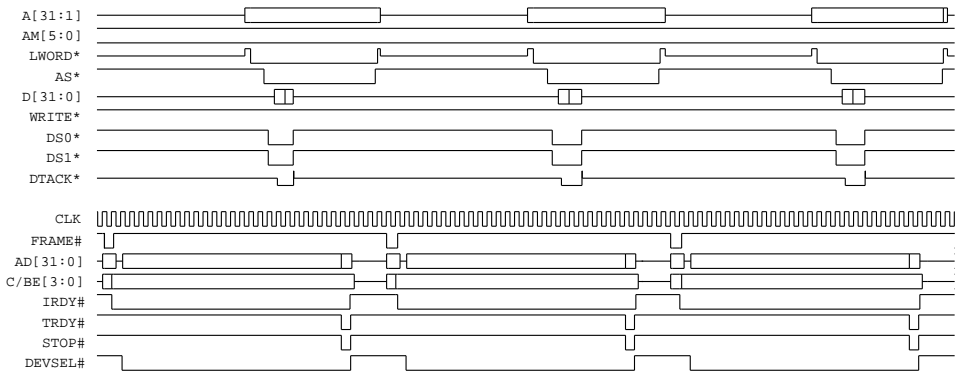
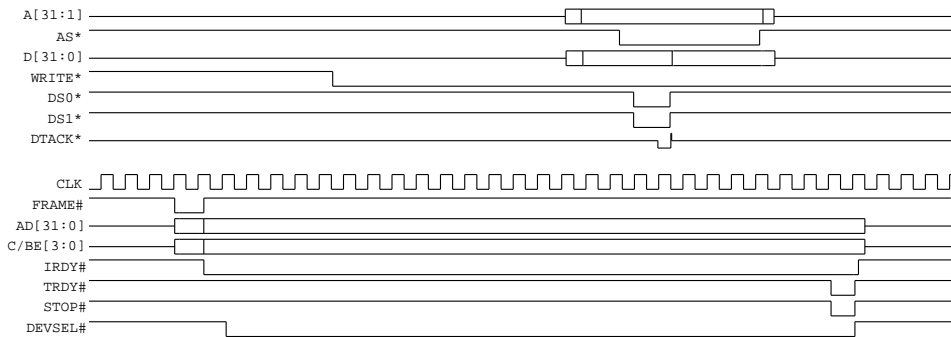


Figure B.2 : Several Coupled Read Cycles - Universe as VME Master

B.1.1.3 Write Cycles

The performance of coupled write cycles is similar to that of coupled read cycles except that an extra wait state is inserted. Figure B.3 shows a coupled write cycle against an ideal VME slave. Twenty-five wait states are inserted on the PCI bus by the Universe before it responds with TRDY#. A slower VME slave response translates directly to more wait states on the PCI bus.

The sustained performance, when generating write cycles from a 32-bit PCI bus against an ideal VME slave is approximately 4 MB/s.



**Figure B.3 : Coupled Write Cycle - Universe as VME Master**

## B.1.2 Decoupled Cycles

### B.1.2.1 Write Cycles

#### Effect of the PCI Aligned Burst Size

The Universe permits enqueueing of PCI write transactions of length up to the value indicated by the PABS bit in the MAST\_CTL register. This bit forces the Universe to break up PCI transactions at a 32-byte or 64-byte aligned address. The value chosen by the designer will depend upon system specific parameters. The higher limit typically gives higher performance in systems capable of generating burst of that size because it permits more data to be transferred with less addressing and arbitration overhead. On the VMEbus, longer block lengths are generated with the accompanying limit on the number of address phases being generated.

It is important to recognize the limits, however, of using a larger burst size. The Universe's TXFIFO is capable of storing a maximum of 128 bytes with a maximum of four transactions. Attempts to enqueue data in the FIFO encounter a target-retry from the Universe if there is insufficient room available for a full burst size. This implies that the incidence of retries encountered by a PCI initiator will increase with the length of the programmed burst size. A side effect of this is that the FIFO may empty more often if the latency between retries exceeds the time taken by the Universe to dequeue a transaction to the VMEbus.

In general, the PABS should be programmed with the size of the burst the Universe will most typically encounter in the system. If the majority of accesses to the TXFIFO will be short transactions, it is recommended that the PABS be programmed for a value of 32 bytes. Only in systems where transactions tend towards the 64-byte length, or where accesses to the TXFIFO are infrequently made should the PABS be programmed to 64 bytes.

## Effect of the PWON Counter

The Posted Write On Counter (PWON in the MAST\_CTL register) controls the maximum tenure that the PCI Slave Channel will have on the VMEbus. Once this channel has gained ownership of the VMEbus for use by the TXFIFO, it only relinquishes it if the FIFO becomes empty or if the number of bytes programmed in the counter expires. In most situations, the FIFO empties before the counter expires. However, if a great deal of data is being transferred by a PCI initiator to the VMEbus, then this counter ensures that only a fixed amount of VME bandwidth is consumed.

Limiting the size of the PWON counter imposes greater arbitration overhead on data being transferred out from the FIFO. This is true even when programmed for ROR mode since an internal arbitration cycle will still occur. The value for the PWON counter must be weighed from the system perspective with the impact of imposing greater latency on other channels (the DMA and Interrupt Channels) and other VME masters in gaining ownership of the VMEbus. On a Universe equipped card which is only performing system control functions, the counter would be set to minimum. On a card which is responsible for transferring considerable amounts of performance-critical data the counter will be set much higher at the expense of system latency.

## PCI Target Response

As the PCI target during decoupled write operations to the VMEbus, the Universe responds in one of two manners:

1. It immediately issues a target retry because the FIFO does not have sufficient room for a full PCI transaction, where a full transaction length is defined by PCI Aligned Burst Size (PABS) field.
2. It responds as a zero-wait state target receiving up to 32 or 64 bytes in a transaction. Once a full transaction has been enqueued, a target-disconnect is issued.

In either case, the Universe will consume the minimum possible PCI bandwidth, never inserting wait states.

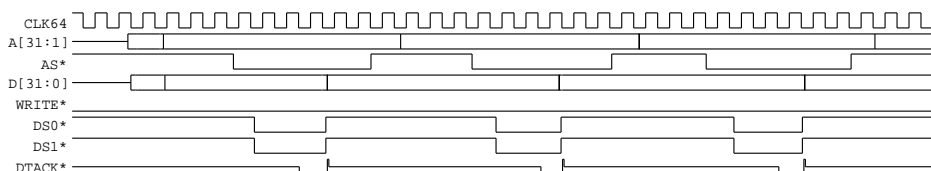
## VME Master Performance

As a VME master, the Universe waits until a full transaction has been enqueued in the FIFO before requesting the VMEbus and generating a VME cycle. If the VMEbus is already owned by the decoupled path (see “Effect of the PWON Counter” on page B-5), the Universe still waits until a full transaction is enqueued in the FIFO before processing it.

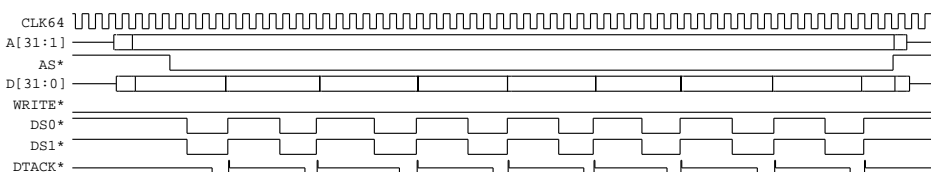
If configured to generate non-block transfers, the Universe can generate back-to-back VME transfers with cycle times of approximately 165ns (AS\* to AS\*) against an ideal VME slave (30-45 ns). A greater cycle time is required between the termination of one full enqueued transaction and the start of the next. This inter-transaction time is approximately 220ns. As such,

the longer the PCI transaction, the greater the sustained performance on the VMEbus. With 64-byte PCI transactions, the sustained rate is 23 MB/s. With 32-byte transactions, this drops to 22 bytes. Each of these numbers is calculated with no initial arbitration or re-arbitration for the bus. Figure B.4 shows the Universe dequeuing a transaction with multiple non-block VME transfers.

Block transfers significantly increase performance. The inter-transaction period remains at approximately 240 ns for BLTs and 280ns for MBLTs, but the data beat cycle time (DS\* to DS\*) drops to about 105ns against the same ideal slave. Again the length of the burst size affects the sustained performance because of the inter-transaction time. For BLTs operating with a burst size of 64 bytes, the sustained performance is 33 MB/s, dropping to 29 MB/s for a burst size of 32 bytes. MBLTs operating with 64-byte bursts perform at a sustained rate of 51 MB/s, dropping to 39 MB/s for 32 bytes.



**Figure B.4 : Several Non-Block Decoupled Writes - Universe as VME Master**



**Figure B.5 : BLT Decoupled Write - Universe as VME Master**

## B.2 VME Slave Channel

### B.2.1 Coupled Cycles

#### B.2.1.1 Block vs. non-Block Transfers

The Universe VME Slave Channel handles both block and non-block coupled accesses in similar manners. Each data beat is translated to a single PCI transaction. Once the transaction has been acknowledged on the PCI bus, the Universe asserts DTACK\* to terminate the VME data beat.

A non-block transfer and the first beat of a BLT transfer have identical timing. In each, the Universe decodes the access and then provides a response to the data beat. Subsequent data beats in the BLT transfer are shorter than the first due to the fact that no address decoding need be performed in these beats.

MBLT transfers behave somewhat differently. The first beat of an MBLT transfer is address only, and so the response is fairly fast. Subsequent data beats require acknowledgment from the PCI bus. With a 32-bit PCI bus, the MBLT data beat (64 bits of data) requires a two data beat PCI transaction. Because of this extra data beat required on the PCI bus, the slave response of the Universe during coupled MBLT cycles is at least one PCI clock greater (depending upon the response from the PCI target) than that during BLT cycles.

#### B.2.1.2 Read Cycles

During coupled cycles, the Universe does not acknowledge a VME transaction until it has been acknowledged on the PCI bus. Because of this the VME slave response during coupled reads is directly linked to the response time for the PCI target. Each clock of latency in the PCI target response translates directly to an extra clock of latency in the Universe's VME coupled slave response.

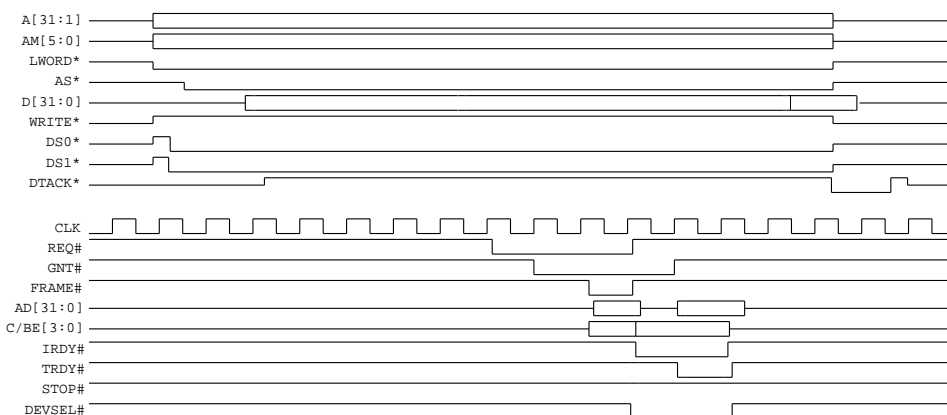
The address of an incoming VME transaction is decoded and translated to an equivalent PCI transaction. From initial assertion of AS\* on the VMEbus to assertion of REQ# on the PCI bus is typically 7-8 PCI clock periods. During the data only portion of subsequent beats in block transfers, the time from DS\* assertion to REQ# is about 6 clocks.

From assertion of REQ#, the Universe does not insert any extra wait states in its operations as an initiator on the PCI bus. Upon receiving GNT# asserted, the Universe asserts FRAME# in the next clock and after the required turn-around phase, asserts IRDY# to begin data transfer.

Once TRDY# is sampled asserted, the Universe responds back to the VMEbus by asserting DTACK\*. If the initiating VME transaction is 64-bit and the PCI bus or PCI bus target are 32 bit, then two data transfers are required on PCI before the Universe can respond with DTACK\*. No wait states are inserted by the Universe between these two data beats on PCI. The assertion

of DTACK\* from the assertion of TRDY# has a latency of 2 clocks. Figure B.6 shows a typical non-block coupled read cycle.

When accessing a PCI target with a zero wait state response, the Universe VME response becomes approximately 14 PCI clock periods (about 420ns in a 33MHz system) during single cycles, and the first beat of a BLT. During pure data beats in both BLT and MBLTs, the slave response becomes 13 clocks.



**Figure B.6 : Coupled Read Cycle - Universe as VME Slave**

### B.2.1.3 Write Cycles

Coupled writes in the VME Slave Channel operate in a similar fashion to the coupled reads. The VME slave response is directly linked to the response of the PCI target. In generating the request to the PCI bus, coupled write cycles require one further clock over reads. Hence, during single cycles, or the first beat of a BLT, the time from AS\* to REQ# asserted is 8-9 PCI clocks, while DS\* to REQ# is 7 clocks for the data beat portion of a block transfer.

As with reads, the response from the PCI target's assertion of TRDY# to DTACK\* assertion by the Universe adds two clocks to the transfer. Figure B.7 shows a typical non-block coupled write cycle.

Because write cycles on the PCI bus require one less clock than reads, due to the absence of the turn-around phase between address and data phases, the overall slave response during coupled writes works out to the same as coupled reads against an identical target. In accessing a zero-wait state PCI target, the Universe's coupled write slave response then is approximately 14 PCI clocks. During subsequent data beats of a block transfer (either BLT or MBLT), the slave response (DS\* to DTACK\*) is 13 clocks.

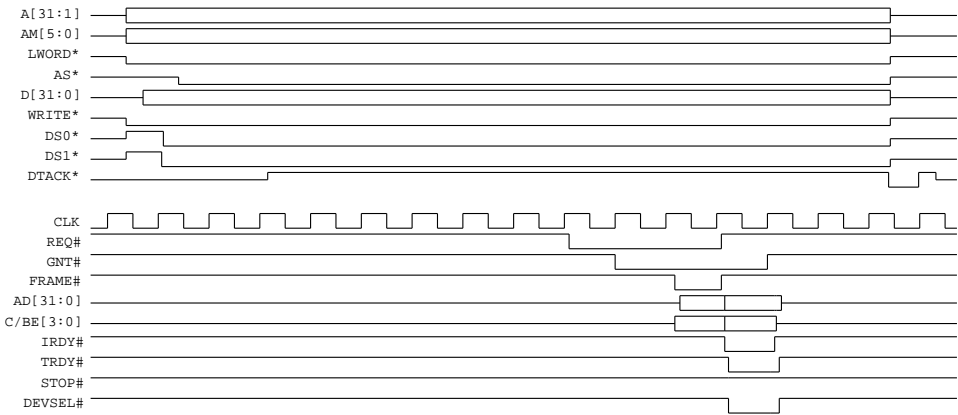


Figure B.7 : Coupled Write Cycle - Universe as VME Slave

B.2.2 Decoupled Cycles

B.2.2.1 Write Cycles

Effect of the PCI Aligned Burst Size

The PCI Aligned Burst Size (PABS in the MAST\_CTL register) affects the maximum burst size that the Universe generates onto the PCI bus; either 32 or 64 bytes. Note that the VME Slave Channel only generates PCI bursts in response to incoming block transfers.

The greater burst size means less arbitration and addressing overhead. However, incumbent in this is the greater average latency for other devices in the PCI system. Hence, in the VME Slave Channel, the burst size is a trade-off between performance and latency.

VME Slave Response

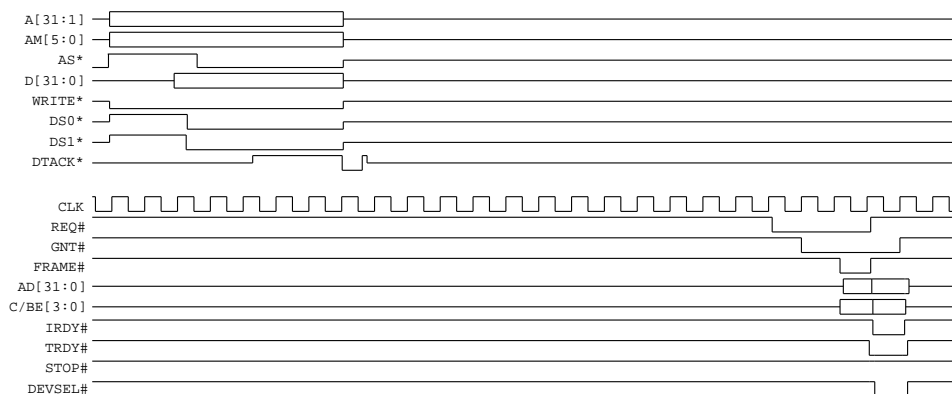
As a VME slave, the Universe accepts data into its RXFIFO with minimum delay provided there is room in the FIFO for a further data beat. Assertion of DTACK\* is delayed if there is insufficient room in the FIFO for the next data beat.

During non-block transfers, the Universe must both decode the address and enqueue the data before asserting DTACK\* to acknowledge the transfer. Because of this, the slave response during non-block transfers is considerably slower than block transfers. This slave response time is 130ns.

During BLT transfers, the slave response in the first data beat being both address decode and data transfer is the same as a non-block transfer, i.e., 130ns. Subsequent data beats, however, are much faster. Response time for these is 60 to 65ns.

During MBLT transfers, the first phase is address only and the slave response is 130ns. Subsequent phases are data only and so the slave response is the same as with BLTs i.e., 60 to 65ns.

Note that the slave response is independent of the data size. D16 non-block transfers have a slave response identical to D32. BLT data beats have slave responses identical to MBLT data beats.



**Figure B.8 : Non-Block Decoupled Write Cycle - Universe as VME Slave**



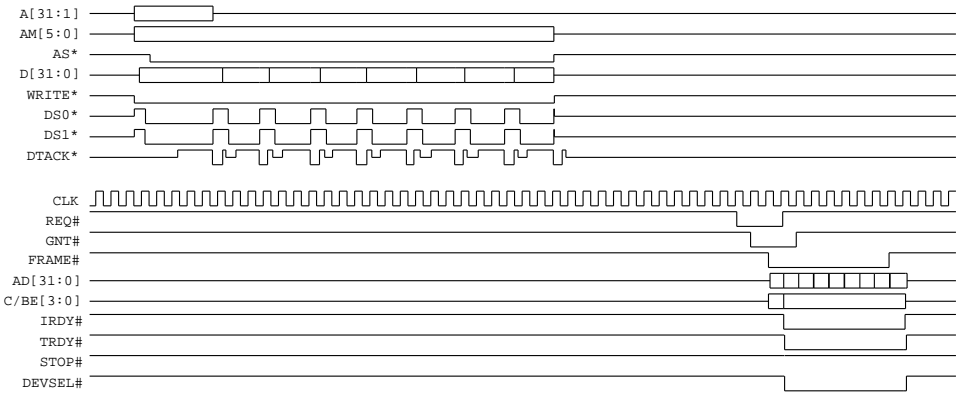


Figure B.9 : BLT Decoupled Write Cycle - Universe as VME Slave

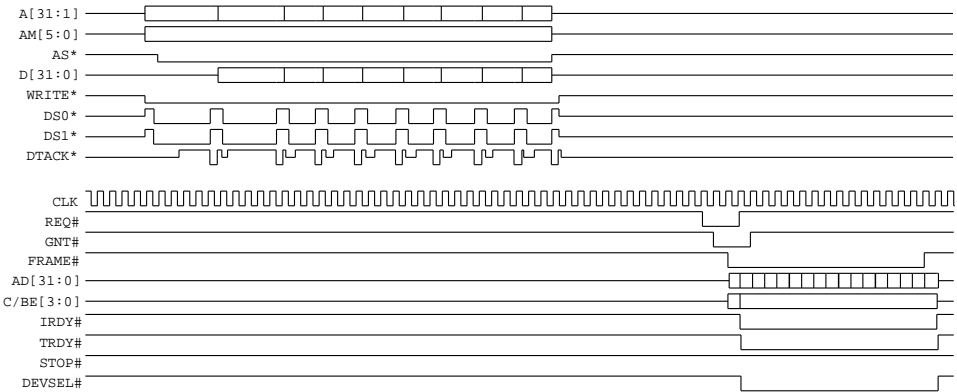


Figure B.10 : MBLT Decoupled Write Cycle - Universe as VME Slave

PCI Master Performance

On the PCI bus, the Universe dequeues data from the RXFIFO once a full VME transaction has been enqueued or once sufficient data has been enqueued to form a PCI transaction of length defined by the PABS field.

Since the Universe does not perform any address phase deletion, non-block transfers are dequeued from the FIFO as single data beat transactions. Only block transfers result in multi-data beat PCI transactions; typically 4 or 8 data beats. In either case, the Universe does not insert any wait states as a PCI master. The clock, after the bus has been granted to the Universe,

drives out FRAME# to generate the address phase. The data phases begin immediately on the next clock. If there is more than one data phase, each phase will immediately follow the acknowledgment of the previous phase.

In each case, because of the lack of any wait states as a PCI master, the Universe is consuming the minimum possible bandwidth on the PCI bus, and data will be written to the PCI bus at an average sustained rate equal to the rate at which the VME master is capable of writing it.

The sustained performance on the PCI bus performing single data beat write transactions to a 32-bit PCI bus is 12 MB/s; double this for a 64-bit bus. When performing 32-byte transactions the sustained performance increases to 76 MB/s; 97 MB/s with 64-byte transactions. Again, these can be doubled for a 64-bit PCI bus. Bear in mind that the PCI bus can only dequeue data as fast as it is being enqueued on the VMEbus. Hence, as the FIFO empties, the sustained performance on the PCI will drop down to match the lower performance on the VME side. However, even with the decreased sustained performance, the consumed bandwidth will remain constant (no extra wait states are inserted while the Universe is master of the PCI bus.)

These numbers assume the PCI bus is granted to the Universe immediately and that the writes are to a zero-wait state PCI target capable of accepting the full burst length. Figure B.1 through Figure B.10 show the Universe responding to non-block, BLT and MBLT write transactions to a 32-bit PCI bus.

### B.2.2.2 Prefetched Read Cycles

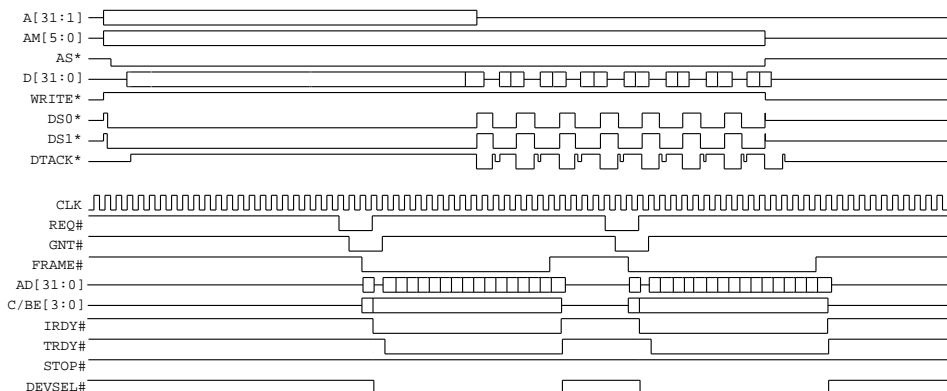
To minimize its slave response, the Universe generates prefetched reads to the PCI bus in response to BLT and MBLT reads coming in from the VMEbus. This option must first be enabled on a per image basis.

When enabled, the Universe will respond to a block read by performing burst reads on the PCI bus of length defined by the PCI Aligned Burst Size (PABS in the MAST\_CTL register). These burst reads continue while the block transfer is still active on the VMEbus (AS\* not negated) and there is room in the FIFO. If there is insufficient room in the FIFO to continue (a common occurrence since the Universe is capable of fetching data from the PCI bus at a much faster rate than a VME master is capable of receiving it), then pre-fetching stops and only continues once enough room exists in the FIFO for another full burst size.

The first data beat of a block transfer must wait for the first data beat to be retrieved from the PCI bus—this is essentially a coupled transfer. See the section on coupled transfers for details on coupled performance. However, once the pre-fetching begins, data is provided by the Universe in subsequent data beats with a slave response of 65ns. This continues while there is data in the FIFO. If the FIFO empties because data is being fetched from the PCI bus too slowly, wait states are inserted on the VMEbus awaiting the enqueueing of more data.

On the PCI bus, the Universe fetches data at 66 MB/s with PABS set to 32-byte transactions; 89 when set to 64-byte transactions. Once the FIFO fills, pre-fetching slows to match the rate at which it is being read from the VMEbus. Bandwidth consumption, however, remains con-

stant, only the idle time between transaction increases.



**Figure B.11 : BLT Pre-fetched Read Cycle - Universe as VME Slave**

## B.3 DMA Channel

### B.3.1 VMEbus Ownership Modes

The DMA has two timers that control its access to the VMEbus: the VON (VMEbus On) counter and the VOFF (VMEbus Off) timer. The VON counter controls the number of bytes that are transferred by the DMA during any VMEbus tenure, while the VOFF timer controls the period before the next request after a VON time-out.

While the bus is more optimally shared between various masters in the system, and average latency drops as the value programmed for the VON counter drops, the sustained performance of the DMA also drops. The DMA is typically limited by its performance on the VMEbus. As this drops off with greater re-arbitration cycles, the average VMEbus throughput will drop. Even if the Universe is programmed for ROR mode, and no other channels or masters are requesting the bus, there will be a period of time during which the DMA will pause its transfers on the bus.

An important point to consider when programming these timers is the more often the DMA relinquishes its ownership of the bus, the more frequently the PCI Slave Channel will have access to the VMEbus. If DMA tenure is too long, the TXFIFO may fill up causing any further accesses to the bus to be retried. In the same fashion, all coupled accesses will be retried while

the DMA has tenure on the bus. This can significantly affect transfer latency and should be considered when calculating the overall system latency.

### **B.3.2 VME Transfers**

On the VMEbus, the Universe can perform D08 through D64 transactions in either block or non-block mode. The time to perform a single beat, however, is independent of the bus width being used. Hence, a D08 transaction will transfer data at 25% the rate of a D32, which in turn is half that for D64.

There is a significant difference between the performance for block vs. non-block operations. Because of the extra addressing required for each data transfer in non-block operations, the DMA performance is about half that compared to operating in block mode. When considering that most VME slaves also have considerably slower slave response in non-block mode, the overall performance may drop to one-quarter of that achievable in block mode.

#### **B.3.2.1 Read Transfers**

When performing non-block reads on the VMEbus, the Universe cycle time (AS\* to next AS\*) is approximately 160ns, which translates to about 25 MB/s when performing D32 transfers. For block transfers the cycle time (DS\* to next DS\*) falls to about 130ns, or 30 MB/s for D32 transfers. For multiplexed block transfers (MBLTs) the cycle time remains the same, but because the data width doubles, the transfer rate increases to about 60MB/s.

#### **B.3.2.2 Write Transfers**

Non-block writes to the VMEbus occur at the same rate as non-block reads: 160ns cycle time (AS\* to next AS\*), or 25MB/s during D32 transfers. Block writes, however, are significantly faster with a 95ns cycle time (DS\* to next DS\*), or 42 MB/s. Multiplexed block transfers have slightly longer cycle times at about 110ns (DS\* to next DS\*), or 72 MB/s with D64 MBLTs.

### **B.3.3 PCI Transfers**

As a master on the PCI bus, the Universe DMA follows the same general set of rules as the VME Slave channel does: it never inserts any wait states into the transfer (i.e., it never negates IRDY# until the transaction is complete) and will always, where possible, generate full aligned bursts as set in the PABS field of the MAST\_CTL register.

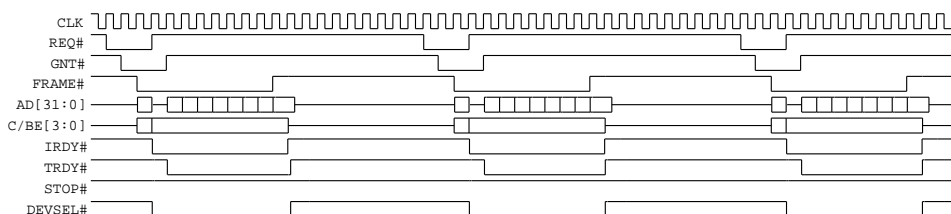
Between transactions on the PCI bus, the Universe DMA typically sits idle for 10 clocks. Hence, minimizing the number of idle periods and re-arbitration times by setting PABS to its maximum value of 64 bytes increases the performance of the DMA on this bus. However, due to limitations in the DMA FIFO size, this does not always result in higher DMA throughput.

The DMA FIFO is only sufficiently large enough to contain two 64-byte PCI transactions. Also, the DMA only generates a read transaction on the PCI bus if there is sufficient room for a full transaction, and a write transaction only if there is a full transaction in the FIFO. A situation could arise during DMA reads (writes to the PCI bus) that the DMA has enqueued a full transaction into the FIFO. At this point, DMA will make an attempt to dequeue that transaction to the PCI bus. If, before the DMA is given an opportunity to dequeue that transaction another is enqueued from the VMEbus, the DMA will give up ownership of the VMEbus. This sort of operation will result in an oscillating tendency for the Universe to perform short transfers on the VMEbus and then release ownership. The effect may be to severely limit the throughput with excessive arbitration overhead.

The same situation could occur during DMA writes (reads from the PCI bus). The DMA begins writes to the VMEbus as soon as a full transaction has been enqueued in the FIFO and releases the bus if the FIFO goes empty, or only a partial transaction has been enqueued. It is possible that the DMA may write the full transaction to the VMEbus before another transaction has been completely enqueued from the PCI side, resulting in a loss of VMEbus ownership by the DMA since it gives up the bus if the FIFO does not contain a full transaction to dequeue.

Encountering either of the above situations depends upon factors such as the VME slave response, the PCI target response and PCI arbitration latency. It may typically be rectified by changing the PABS field. A smaller burst size makes it more likely that a full transaction will always be in the FIFO during writes to the VMEbus. A larger burst size allows the FIFO to empty more quickly during writes to the PCI bus.

Although both read and write transactions occur on the PCI bus with zero wait states, there is a period of eight PCI clocks during which the Universe remains idle before re-requesting the bus for the next transaction. With PABS set for 32-byte transactions on a 32-bit PCI bus, this translates to a peak transfer rate of 51 MB/s for reads, 53 MB/s for writes, doubling to 102 and 106 for a 64-bit PCI bus. With PABS set for 64-byte transactions, the peak transfer rate increases to 74 MB/s for reads, 77 MB/s for writes on a 32-bit PCI bus, doubling to 148 MB/s and 154 MB/s respectively for 64-bit PCI buses.



**Figure B.12 : PCI Read Transactions During DMA Operation**

## B.4 Summary

**Table B.1 : PCI Slave Channel Performance**

Cycle Type	Performance
Coupled Read - PCI target response	24 PCI clocks
Coupled Write - PCI target response	25 PCI clocks
Decoupled Write - non-block D32 - VME cycle time - sustained perf (PABS =0) - sustained perf (PABS =1) - D32 BLT - VME cycle time - sustained perf (PABS =0) - sustained perf (PABS =1) - D64 MBLT - VME cycle time - sustained perf (PABS =0) - sustained perf (PABS =1)	165 ns 22 MB/s 23 MB/s 105 ns 29 MB/s 33 MB/s 105 ns 39 MB/s 51 MB/s
Register access	8 wait states

**Table B.2 : VME Slave Channel Performance**

Cycle Type	Performance
Coupled Read <ul style="list-style-type: none"><li>- non-block</li><li>- block</li></ul>	420 ns 390 ns
Coupled Write <ul style="list-style-type: none"><li>- non-block</li><li>- block</li></ul>	420 ns 390 ns
Pre-fetched Read <ul style="list-style-type: none"><li>- VME slave response (1st data beat)</li><li>- VME slave response (other data beats)</li></ul>	(see coupled table) 65 ns
Decoupled Write <ul style="list-style-type: none"><li>- non-block slave response</li><li>- block slave response</li></ul>	130 ns 65 ns
Register access <ul style="list-style-type: none"><li>- reads</li><li>- writes</li></ul>	360 ns 240 ns

**Table B.3 : DMA Channel Performance**

Cycle Type	Performance
PCI Reads - PABS = 32 bytes - PABS = 64 bytes	51 MB/s 74 MB/s
PCI Writes - PABS = 32 bytes - PABS = 64 bytes	53 MB/s 77 MB/s
VME Reads - non-block D32 - D32 BLT - D64 MBLT	25 MB/s 30 MB/s 60 MB/s
VME Writes - non-block D32 - D32 BLT - D64 MBLT	25 MB/s 42 MB/s 72 MB/s

**Table B.4 : Daisy Chains**

Daisy Chain	Performance
IACK Daisy Chain - active Universe interrupt - no active Universe interrupt	21 - 36 ns 6 - 21 ns
Bus Grant Daisy Chain	32 ns



## Appendix C Typical Applications

Being a bridge between standard interfaces, the Universe requires minimal external logic to interface to either the VMEbus or to the PCI bus. In most applications, only transceivers to buffer the Universe from the VMEbus, plus some reset logic are all that is required. The following information should be used only as a guide in designing the Universe into a PCI/VME application. Each application will have its own set-up requirements.

### C.1 VME Interface

#### C.1.1 Transceivers

The Universe has been designed such that it requires full buffering from VMEbus signals. Necessary drive current to the VMEbus is provided by the transceivers while at the same time isolating the Universe from potentially noisy VMEbus backplanes. In particular, complete isolation of the Universe from the VMEbus backplane allows use of ETL transceivers which provide high noise immunity as well as use in live insertion environments. The VME community is presently working on a specification “VME64 Extensions” which provides migration of VME systems to live insertion environments.

If neither live insertion nor noise immunity are a concern, those buffers that provide input only (U15 and U17 in Figure C.1, below) may be omitted. The daisy chain input signals, BGIN[3:0] and IACKIN, have Schmitt trigger inputs, which should rectify any minor noise on these signals. If considerable noise is expected, the designer may wish to put external filters on these signals. Bear in mind that any filtering done on these signals will detrimentally affect the propagation of bus grants down the daisy chain. Only extremely noisy systems or poorly designed backplanes should require these filters.

Figure C.1 shows one example of how to connect the Universe to the VMEbus. The transceivers in this example were chosen to meet the following criteria:

- provide sufficient drive strength as required by the VME specification (see Table C.1 on page C-4),
- meet Universe skew requirements, and
- minimize part counts.

U15 and U17 in Figure C.1 are optional devices. They will provide better noise immunity.

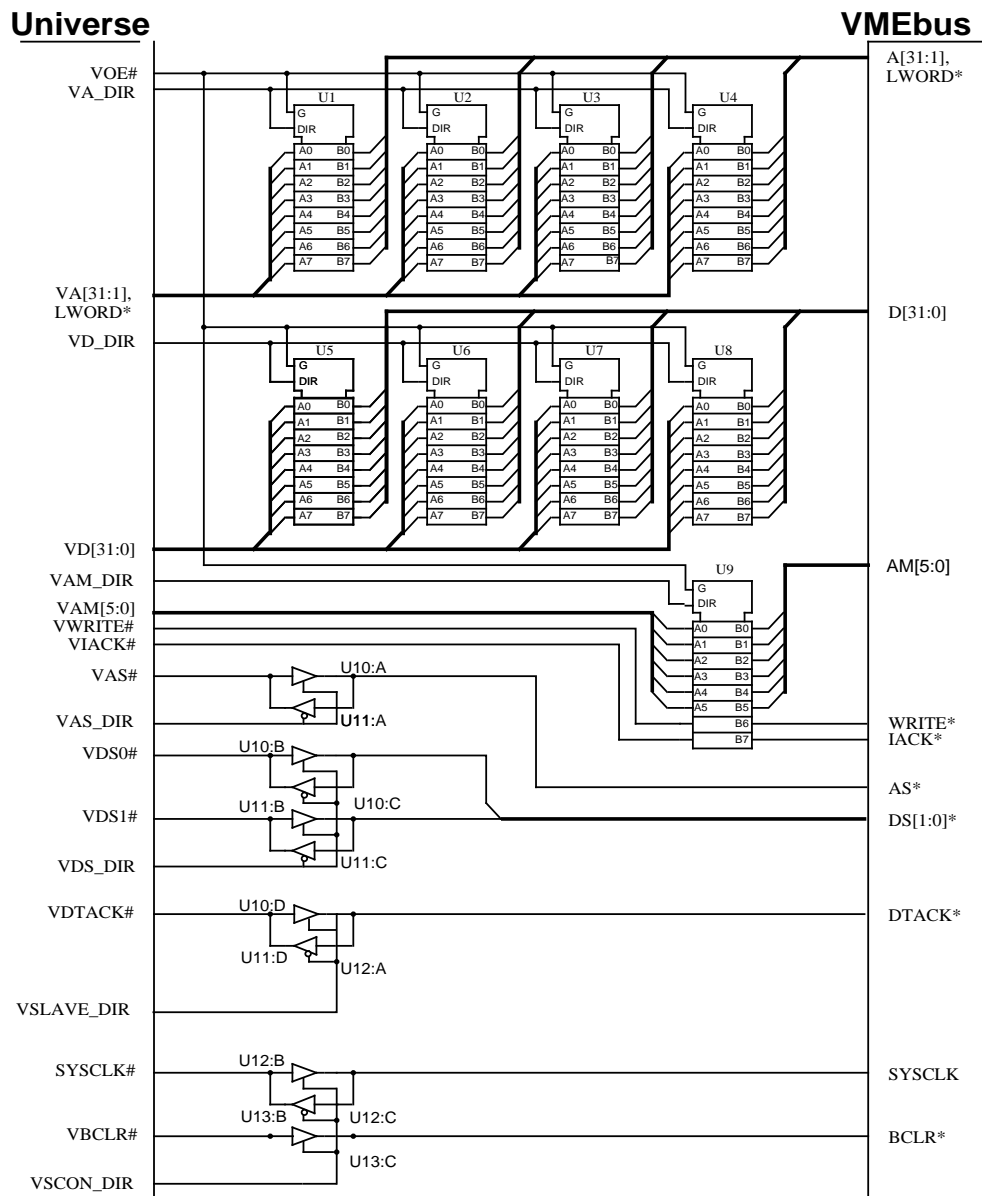
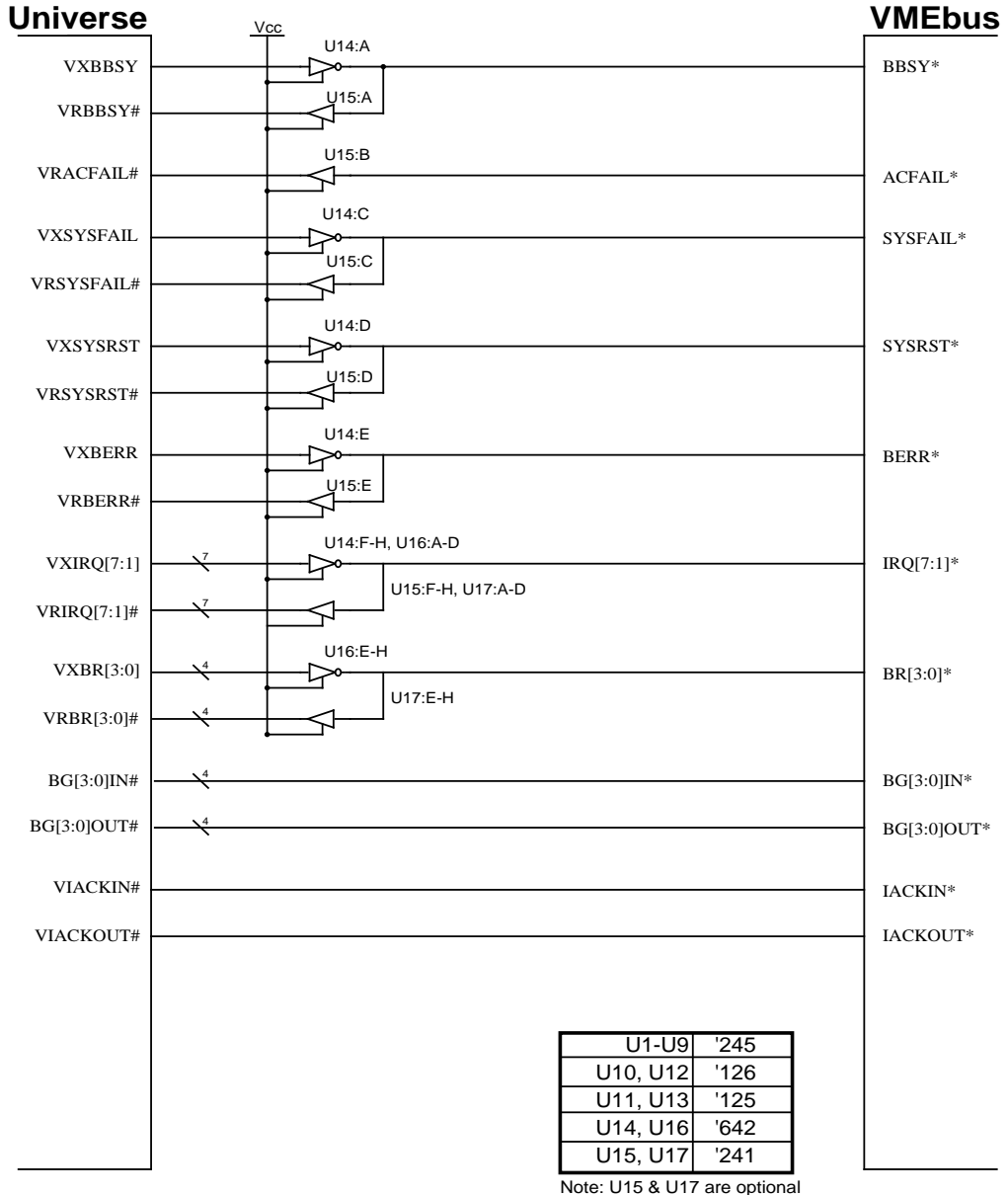


Figure C.1 : Universe Connections to the VMEbus Through TTL Buffers



**Figure C.1 : Universe Connections to the VMEbus Through TTL Buffers  
(continued)**

F series transceivers were chosen for this example, but a faster family, such as ABT, may also be used. Care should be taken in the choice of transceivers to avoid ground bounces and also to minimize crosstalk incurred during switching. To limit the effects of crosstalk, the amount of routing under these transceivers must be kept to a minimum. Daisy chain signals can be especially susceptible to crosstalk.

**Table C.1 : VMEbus Signal Drive Strength Requirements**

VME bus Signal	Required Drive Strength
A[31:1], D[31:0], AM[5:0], IACK*, LWORD*, WRITE*, DTACK*	IOL ≥ 48mA IOH ≥ 3mA
AS*, DS[1:0]*,	IOL ≥ 64mA IOH ≥ 3mA
SYSCLK*	IOL ≥ 64mA IOH ≥ 3mA
BR[3:0]*, BSY*, IRQ[7:0]*, BERR*, SYSFAIL*, SYSRESET*	IOL ≥ 48mA

### C.1.1.1 Pull down resistors

The Universe has internal pull down resistors which are used for its default power-up option state. (The pins requiring pull-ups or pull-downs are indicated in Table 4.2 on page 4-3). These internal pull down resistors, ranging from 25kΩ-500kΩ, are designed to sink between 10μA-200μA. F-series buffers, however, can source up to 650 μA of current (worst case). This sourced current has the ability to override the internal power up resistors on the Universe. This may cause the Universe to incorrectly sample a logic "1" on the pins. To counteract this potential problem, assuming a worst case scenario of a 650 μA current, Tundra recommends connecting a 1K resistor to ground, in parallel, with the internal pull down resistor.

Tundra recommends that any pins controlling the powerup options which are critical to the application at powerup be connected to ground with a pull down resistor as described above. If these options are not critical and if it is possible to reprogram these options after reset, additional resistors need need not be added. Note that all options set by pins left in a default state without external pull down resistors should be reprogrammed after reset to ensure that they are correctly initialized.

### C.1.2 Direction control

When the Universe is driving VMEbus lines, it drives the direction control signals high (i.e., VA\_DIR, VAM\_DIR, VAS\_DIR, VD\_DIR, VDS\_DIR, VSLAVE\_DIR, and VSCON\_DIR). When the VMEbus is driving the Universe, these signals are driven low. The control signals in the Universe do not all have the same functionality. Since the Universe implements early bus release, VAS\_DIR must be a separate control signal.

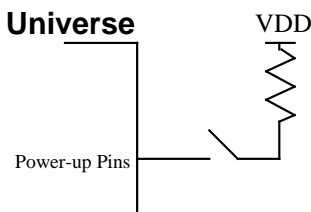
Contention between the Universe and the VME buffers is handled since the Universe tristates its outputs one 64MHz clock period before the buffer direction control is faced inwards.

### C.1.3 Power-up Options

Power-up options for the automatic configuration of slave images and other Universe features are provided through the state of the VME address and data pins, VA[31:1] and VD[31:27]. All of these signals are provided with internal pull-downs to bias these signals to their default conditions. Should values other than the defaults be required here, either pull-ups or active circuitry may be applied to these signals to provide alternate configurations.

Since the power-up configurations lie on pins that may be driven by the Universe or by the VME transceivers, care must be taken to ensure that there is no conflict. During any reset event, the Universe does not drive the VA or VD signals. As well, during any VMEbus reset (SYSRST\*) and for several CLK64 periods after, the Universe negates VOE# to tri-state the transceivers. During this period that these signals are tri-stated, the power-up options are loaded with their values latched on the rising edge of CLK64 before VOE# is re-asserted.

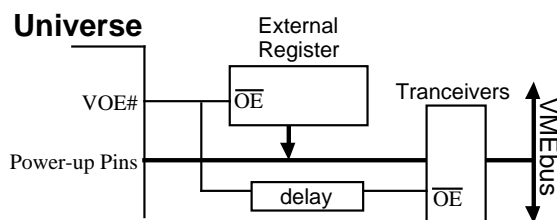
Configuration of power-up options is most easily accomplished through passive 10k pull-up resistors on the appropriate VA and VD pins. The configurations may be made user-configurable through jumpers or switches as shown in Figure C.2



**Figure C.2 : Power-up Configuration Using Passive Pull-ups**

Alternatively, an active circuit may be designed which drives the VA and VD pins with pre-set (or pre-programmed) values. This sort of circuit would be of value when power-up configurations such as the register access slave image are stored in an external programmable register. To implement this circuit, the VOE# output from the Universe must be monitored. When the Universe negates this signal, the appropriate VA and VD signals may be driven and

upon re-assertion the drive must be removed. To avoid conflict with the transceivers, logic must be designed such that the enabling of the transceivers does not occur until some point after the configuration options have been removed from the VD and VA signals. Figure C.3 shows one such implementation. The delay for enabling of the VMEbus transceivers could be implemented through clocked latches.



**Figure C.3 : Power-up Configuration Using Active Circuitry**

### Auto-Syscon and PCI Bus Width Power-up Options

The VME64 specification provides for automatic enabling of the system controller in a VME system through monitoring of the BGIN3\* signal. If at the end of SYSRST\* this pin is low, then the system controller is enabled; otherwise it is disabled. The Universe provides an internal pull-down resistor for this function. If it is in slot one, this pin will be sampled low. If not in slot one, then it will be driven high by the previous board in the system and system controller functions will be disabled. No external logic is required to implement this feature.

## C.2 PCI Bus Interface

The Universe provides a fully standard PCI bus interface compliant for both 32-bit and 64-bit designs. No external transceivers or glue logic is required in interfacing the Universe to any other PCI compliant devices. All signals may be routed directly to those devices.

At reset, the Universe is configured with a 32-bit or 64-bit PCI data bus, depending on the level of REQ64#. The Universe is configured as 32-bit PCI if REQ64# is high; it is configured as 64-bit if REQ64# is low. Accordingly, users must ensure that REQ64# is driven high or low at reset; otherwise, REQ64# will be floating and whether the Universe is reset as a 64-bit or 32-bit PCI is indeterminate. In a 64-bit PCI bus, an external resource drives REQ64# low; this enables the Universe as 64-bit. To configure the device for a 32-bit PCI bus, pull down

REQ64# with a 4.7k $\Omega$  resistor. On a 32-bit PCI bus, the Universe drives all its 64-bit extension bi-direct signals at all times; these signals include: C/BE[7:4]#, AD[63:32], REQ64#, PAR64# and ACK64# to unknown values. Independent of the setting of the LD64EN bit, the Universe will never attempt a 64-bit cycle on the PCI bus if it is powered up as 32-bit.

## C.2.1 Resets

The Universe provides several reset input and outputs which are asserted under various conditions. These can be grouped into three types as shown in Table C.2.

**Table C.2 : Reset Signals**

Group	Signal Name	Direction
VMEbus	VXSYSRST	output
	VRSYSRST#	input
PCI bus	LRST#	output
	RST#	input
	VME_RESET#	input
Power-up	PWRRST#	input

### VMEbus Resets

The VMEbus resets are connected to the VMEbus as indicated in Figure C.1 on page C-2 through external buffers.

### PCI bus Resets

Use of the PCI bus resets will be application dependent. The RST# input to the Universe should typically be tied in some fashion to the PCI bus reset signal of the same name. This will ensure that all Universe PCI related functions are reset together with the PCI bus.

The LRST# pin is a totem-pole output which is asserted due to any of the following initiators:

- PWRRST#,
- VRSYSRST#,
- local software reset (in the MISC\_CTL register), or
- VME CSR reset (in the VCSR\_SET register).

The designer may wish to disallow the Universe from resetting the PCI bus in which case this output may be left unconnected. Otherwise LRST# should be grouped with other PCI reset generators to assert the RST# signal such that:

$RST\# = LRST\# \& \text{reset\_source1} \& \text{reset\_source2} \& \dots$

If the Universe is the only initiator of PCI reset,  $LRST\#$  may be directly connected to  $RST\#$ .

Assertion of  $VME\_RESET$  causes the Universe to assert  $VXSYSRST$ .



*This signal must not be tied to the PCI  $RST\#$  signal unless the Universe  $LRST\#$  output will not generate a PCI bus reset. Connecting both  $LRST\#$  and  $VME\_RESET\#$  to  $RST\#$  will cause a feedback loop on the reset circuitry forcing the entire system into a endless reset.*

To reset the VMEbus through this signal it is recommended that it be asserted for several clock cycles until the Universe asserts  $RST\#$  and then released. This ensures a break is made in the feedback path.

### Power-up Reset

The  $PWRRST\#$  input is used to provide reset to the Universe until the power supply has reached a stable level. It should be held asserted for 100 milliseconds after power is stable. Typically this can be achieved through a resistor/capacitor combination although more accurate solutions using under voltage sensing circuits (i.e. MC34064) are often implemented.

### JTAG Reset

The JTAG reset,  $TRST\#$ , should be tied into the master system JTAG controller. It resets the Universe internal JTAG controller. If JTAG is not being used, this pin should be tied to ground.

## C.2.2 Local Interrupts

The Universe provides eight local bus interrupts, only one of which has drive strength that is fully PCI compliant. If any of the other seven interrupts are to be used as interrupt outputs to the local bus (all eight may be defined as either input or output), an analysis must be done on the design to determine whether the 4 mA of drive that the Universe provides on these lines is sufficient for the design. If more drive is required, the lines may simply be buffered.

All Universe interrupts are initially defined as inputs. To prevent excess power dissipation, any interrupts defined as inputs should always be driven to either high or low. Pull-ups should be used for this purpose rather than direct drive since a mis-programming of the interrupt registers may cause the local interrupts to be configured as outputs and potentially damage the device.



## C.3 Manufacturing Test Pins

The Universe has several signals used for manufacturing test purposes. They are listed in Table 2.22 on page 2-103, along with the source to which they should be tied.

## C.4 Decoupling $V_{DD}$ and $V_{SS}$ on the Universe

This section is intended to be a guide for decoupling the power and ground pins on the Universe. A separate analog power and ground plane is not required to provide power to the analog portion of the Universe. However, to ensure a jitter free PLL operation, the analog  $AV_{DD}$  and  $AV_{SS}$  pins must be noise free. The following are recommended solutions for noise free PLL operation.

The Analog Isolation Scheme consists of the following:

- a  $0.1\mu\text{F}$  capacitor between the  $AV_{DD}$  and  $AV_{SS}$  pins, and
- corresponding inductors between the pins and the board power and ground planes (See Figure C.4). These inductors are not necessary, but they are recommended.

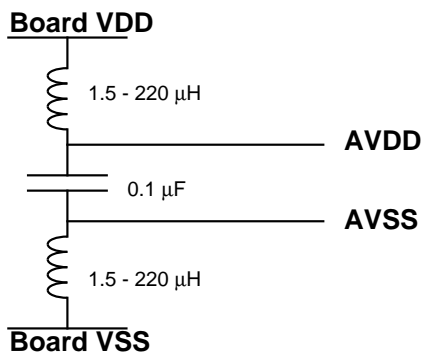
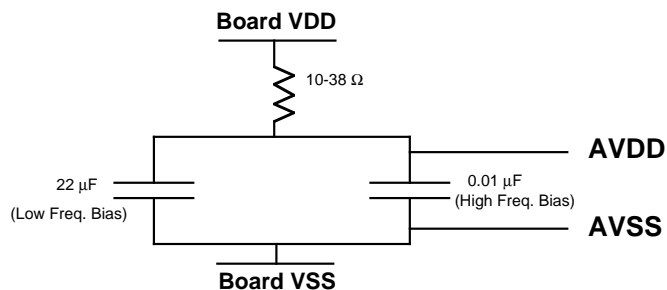


Figure C.4 : Analog Isolation Scheme

The Noise Filter Scheme filters out the noise using two capacitors to filter high and low frequencies (See Figure C.5).

**Figure C.5 : Noise Filter Scheme**

For both schemes, it is recommended that the components involved be tied as close as possible to the associated analog pins.

In addition to the decoupling schemes shown above, it is recommended that  $0.1\ \mu\text{F}$  bypass capacitors should be tied between every three pairs of  $V_{\text{DD}}$  pins and the board ground plane. These bypass capacitors should also be tied as close as possible to the package.

# Appendix D Reliability Prediction

This section is designed to help the user to estimate the inherent reliability of the Universe, as based on the requirements of MIL HDBK217F. This information is recommended for personnel who are familiar with the methods and limitations contained in MIL HDBK217F. The information serves as a guide only; meaningful results will be obtained only through careful consideration of the device, its operating environment, and its application.

## D.1 Physical characteristics

- CMOS gate array
- 100,000 gates (estimate)
- 0.65  $\mu\text{m}$  feature size
- 476 mils x 476 mils scribed die size

## D.2 Thermal characteristics

Idle power consumption:	2.85 Watts
Typical power consumption (32-bit PCI):	3.35 Watts
Maximum power consumption* (32-bit PCI):	4.05 Watts
Typical power consumption (64-bit PCI):	3.75 Watts
Maximum power consumption* (64-bit PCI):	4.65 Watts

\*NOTE: Maximum power consumption is worst case consumption when the Universe is performing DMA reads from the VME bus with alternating worst case data patterns (\$FFFF\_FFFF, \$0000\_0000 on consecutive cycles), and 100pF loading on the PCI bus.

In the majority of system applications, the Universe will consume typical values or less. Typical power consumption numbers are based on the Universe remaining idle 30%-50% of the time, which is significantly less than what is considered likely in most systems. For this reason, it is recommended that typical power consumption numbers be used for power estimation and ambient temperature calculations, as described below.

Reliability calculations of the Universe design in Motorola's H4CPlus Gate Array family show that the Failure In Time (FIT) rate is 68 at a junction temperature of 125°C (maximum junction temperature). (Failure in time is the basic reliability rate expressed as failures per billion (1e-9) device hours. Mean Time Between Failures (MTBF) is the reciprocal of FIT. MTBF is the predicted number of device hours before a failure will occur.)

## D.3 Universe Ambient Operating Calculations

The maximum ambient temperature of the Universe can be calculated as follows:

$$T_a \leq T_j - \theta_{ja} * P$$

Where,

$T_a$  = Ambient temperature (°C)

$T_j$  = Maximum Universe Junction Temperature (°C), i.e., 125°C

$\theta_{ja}$  = Ambient to Junction Thermal Impedance (°C / Watt)

$P$  = Universe power consumption (Watts)

The ambient to junction thermal impedance ( $\theta_{ja}$ ) is dependent on the air flow in linear feet per minute over the Universe. The values for  $\theta_{ja}$  over different values of air flow are as follows:

**Table D.1 :  $\theta_{ja}$  Values For Various Air Flows and Package Types**

Air Flow (LFPM)	0	300	500
313 PBGA	20.10	15.10	14.00
324 CBGA	17.80	13.50	12.30

For example, the maximum ambient temperature of the 313 PBGA, 32-bit PCI environment with 300 LFPM blowing past the Universe is:

$$T_a \leq T_j - \theta_{ja} * P$$

$$T_a \leq 125 - 15.10 * 3.35$$

$$T_a \leq 74.4^\circ\text{C}$$

Hence the maximum rated ambient temperature for the Universe in this environment is 74°C. The thermal impedance can be improved by approximately 10% by adding thermal conductive tape to the top of the packages and through accounting for heat dissipation into the ground planes. This would improve the maximum ambient temperature to 79°C in the above example. Further improvements can be made by adding heat sinks to the PBGA package.

## D.4 Thermal vias

The 313-pin plastic BGA package contains thermal vias which directly pipe heat from the die to the solder balls on the underside of the package. The solder balls use the capabilities of the power and ground planes of the printed circuit board to draw heat out of the package.

# Appendix E Cycle Mapping

The Universe always performs Address Invariant translation between the PCI and VMEbus ports. Address Invariant mapping preserves the byte ordering of a data structure in a little-endian memory map and a big-endian memory map.

## E.1 Little-endian Mode

Table E.1 below shows the byte lane swapping and address translation between a 32-bit little-endian PCI bus and the VMEbus for the address invariant translation scheme.

**Table E.1 : Mapping of 32-bit Little-Endian PCI Bus to 32-bit VMEbus**

PCI Bus						Byte Lane Mapping	VMEbus			
Byte Enables				Address			DS1	DS0	A1	LW
3	2	1	0	1	0					
1	1	1	0	0	0	D0-D7 <=> D8-D15	0	1	0	1
1	1	0	1	0	1	D8-D15 <=> D0-D7	1	0	0	1
1	0	1	1	1	0	D16-D23 <=> D8-D15	0	1	1	1
0	1	1	1	1	1	D24-D31 <=> D0-D7	1	0	1	1
1	1	0	0	0	0	D0-D7 <=> D8-D15	0	0	0	1
						D8-D15 <=> D0-D7				
1	0	0	1	0	1	D8-D15 <=> D16-D23	0	0	1	0
						D16-D23 <=> D8-D15				
0	0	1	1	1	0	D16-D23 <=> D8-D15	0	0	1	1
						D24-D31 <=> D0-D7				
1	0	0	0	0	0	D0-D7 <=> D24-D31	1	0	0	0
						D8-D15 <=> D16-D23				
						D16-D23 <=> D8-D15				
0	0	0	1	0	1	D8-D15 <=> D16-D23	0	1	0	0
						D16-D23 <=> D8-D15				
						D24-D31 <=> D0-D7				

**Table E.1 : Mapping of 32-bit Little-Endian PCI Bus to 32-bit VMEbus** (Continued)

PCI Bus						Byte Lane Mapping	VMEbus			
Byte Enables				Address			DS1	DS0	A1	LW
3	2	1	0	1	0					
0	0	0	0	0	0	D0-D7 <=> D24-D31	0	0	0	0
						D8-D15 <=> D16-D23				
						D16-D23 <=> D8-D15				
						D24-D31 <=> D0-D7				

The unpacking of multiplexed 64-bit data from the VMEbus into two 32-bit quantities on a little-endian PCI bus is outlined in Table E.2 below.

**Table E.2 : Mapping of 32-bit Little-Endian PCI Bus to 64-bit VMEbus**

Byte Enables				Address			PCI to VME Byte Lane Mapping
3	2	1	0	2	1	0	
First Transfer (D32-D63)							
0	0	0	0	0	0	0	D0-D7 <-> A24-A31 (D56-D63)
							D8-D15 <-> A16-A23 (D48-D55)
							D16-D23 <-> A8-A15 (D40-D47)
							D24-D31 <-> LWORD, A1-A7 (D32-D39)
Second Transfer (D0-D31)							
0	0	0	0	1	0	0	D0-D7 <-> D24-D31
							D8-D15 <-> D16-D23
							D16-D23 <-> D8-D15
							D24-D31 <-> D0-D7

# Appendix F

## Why and How to Use BGA Packaging

This section reproduces an article that appears in a 1996 issue of the *Asian Electronics Engineer*.

Just as you were getting used to fine pitch PQFP, out comes a new technology with compelling advantages over PQFP. Ball Grid Arrays (BGAs) are gaining momentum as the preferred high lead-count surface-mount package. You will probably prefer plastic BGA (PBGA) over PQFP because

- BGA is smaller and lighter,
- BGA provides greater assembly yields, and
- BGA improves thermal performance.

Fortunately, unlike your migration to PQFP, your move to BGA will be easier because

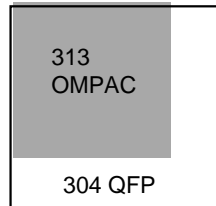
- BGA has comparable electrical performance,
- BGA uses standard, conventional placement and surface mount technology (SMT),
- BGA is compatible with standard SMT processes, and
- the cost of BGA is comparable to similar packaging options, such as QFP.

In this article we summarize the advantages of BGA in relation to other packages, we expose its structure, and we explain how you can use it.

### F.1 Comparing BGA to Other Surface Mount Packages

#### Footprint and height of BGA

A 313-pin BGA uses 34% less of your space than a 304-pin PQFP. This is because while the PQFP only has pins on its perimeter, the BGA can use the entire surface.



**Figure F.1 : Footprint and height of BGA**

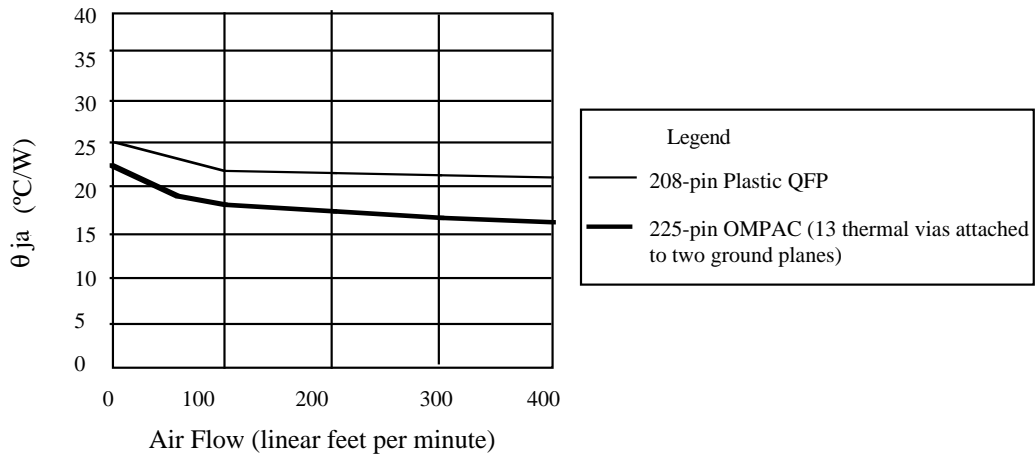
Compare the area of a 313-pin PBGA with a 304-pin QFP. The smaller footprint allows board designers to include more components and greater functionality in their boards.

You will also find that the *mounted profile* of the BGA is less than that of a PQFP. For example, the mounted height of a 313-pin PBGA is approximately 2.1 mm whereas the mounted height of a 304-pin PQFP is approximately 3.7 mm.

### **Thermal characteristics**

As Figure F.1 shows you, the temperature of a BGA package remains significantly closer to the ambient temperature while cooking than the PQFP does. The main advantage of the BGA's low temperature differential is that the die runs cooler. Therefore, you can put the part in a higher ambient temperature environment. You might even be able to eliminate a cooling fan. Finally, you may find that your part lasts longer.

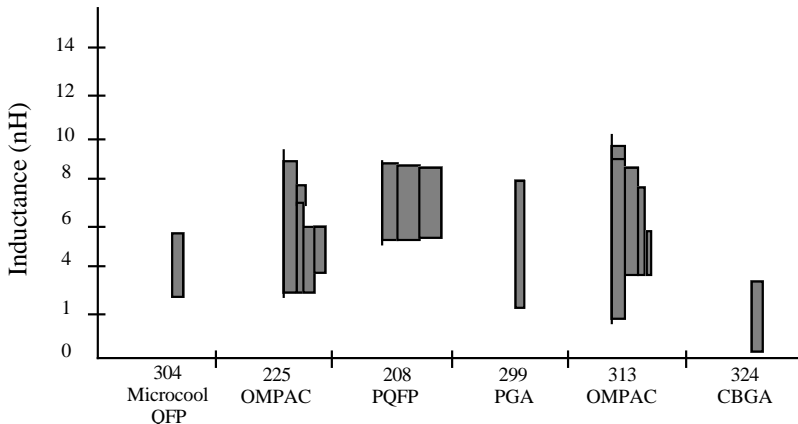




**Figure F.2 : Thermal characteristics of QFP vs. OMPAC**

Figure F.2 shows you  $\theta_{ja}$ , i.e., the difference between the ambient temperature and the temperature at package junctures (given the same wattage), at various air flows, for both packages. You see that the 225-pin OMPAC (a PBGA) compares favorably against the 208-pin PQFP.

### Electrical performance

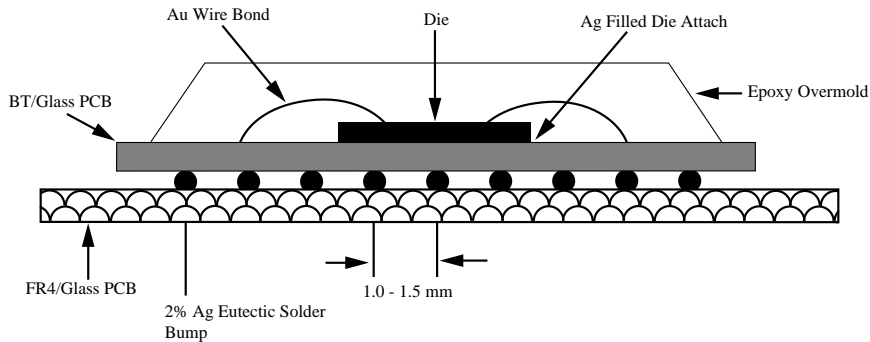


**Figure F.3 : Electrical performance of various packages.**

Figure F.3 shows you the lead inductance of BGA packages in relation to QFP and PGA. Notice that OMPACs are quite comparable to the others.

## F.2 Anatomy of the PBGA

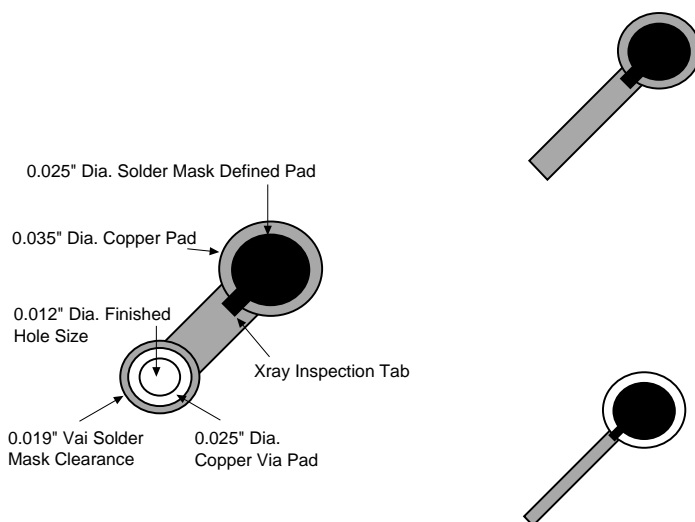
Figure F.4 gives you a cross-section of the PBGA. The die is mounted to a substrate made of Bismaleimide Triazine (BT) epoxy glass laminates. BT is selected due to its relatively high glass transition temperature (170-215°C) compared to FR4's (which is 115-125°C). The higher glass transition temperature gives the package better dimensional stability during the manufacturing process. The substrate usually has two layers, but could contain up to four layers. The manufacturers may add thermal vias and a ground plane to improve both the thermal and electrical characteristics. The die attach material is silver-filled (conductive) epoxy. Both die attachment and die bonding follow conventional techniques. Once the die attach and wire bonding are complete, the die is encapsulated (overmolded) using a plastic mold compound, and conventional transfer molding techniques. For this reason, the package is also called the "Over Molded Pad Array Carrier" (OMPAC). Finally, performed solder spheres (Sn62-Pb36-Ag2) are gang fluxed, placed on the lands on the bottom of the substrate, and reflowed into place.



**Figure F.4 : Cross-Section of the PBGA**

### F.2.1 Bringing the BGA to Your Circuit Board

Figure F.5 below shows you the preferred printed circuit board (PCB) pad design where the solder mask defined pad has an optimum exposed copper pad size of 25 mils (0.65 mm). The determining factor for copper pad diameter is that it should allow for worst case solder mask-to-artwork misregistration. Typically, misregistration tolerance is  $\pm 2$  to  $\pm 4$  mils (although it may be as great as  $\pm 5$  mils). Additional variation in the solder mask opening may arise from overdeveloping and aperture diameter changes. You must ensure that the combined effects of solder mask misregistration and aperture tolerance do not produce an aperture with a diameter less than the solder ball itself. The pad should be 1/2 oz. copper plated to 1.0 oz. It should have a tab. This tab is used to give a solder-wetting indication on X-ray inspection. A solder mask defined pad forms an indent which improves both the placement accuracy and the adhesion of the land to the PCB.



**Figure F.5 : Land Design Patterns**

"Escape routing" is routing the signal, power and ground from beneath the BGA. The level of difficulty you will face here depends on the board technology you are using (i.e., line widths, space widths and the number of board layers) as well as the board space limitations (which affect ball pitch). Regardless of the line and space widths, the outer row of pins can escape onto the PCB device layer. Table F.1 gives an indication of how many rows, besides the outer row, you can route on the top PCB layer for a given board technology, device ball pitch, and pad size.

You should carefully consider the escape method to be used before choosing the geometry of your pad. Escape routing may be onto the PCB device layer, onto another PCB layer by using a via, or a combination of the two.

**Table F.1 : Parameters affecting escape routing**

Board Technology (Line/Space in mm/mil)	Number of Escape Traces Routed Between Pads Given Pitch & Pad Geometry (Pitch/Pad Diameter in mm/mil) <sup>a</sup>			
	(1.5/31)	(1.5/35)	(1.27/31)	(1.27/35)
3/3	4	3	2	2
4/4	3	2	1	1
5/5	2	1	1	1
6/6	1	1	1	
7/7	1	1		
8/8	1	1		
9/9	1			

*a.* When encountered, fractions of traces were rounded down.

For example, apply the following values to the appropriate fields in Table F.1:

- board technology: 6 mil lines and 6 mil spacing
- device ball pitch: 1.27 mm
- 31 mil copper pad with 3 mil older mask clearance around it (this will give you a 25 mil exposed copper pad).

Table F.1 shows that, in these conditions, besides the outer row, you could route-to-escape one more row on the top layer of the PCB.

If you were using integral vias on the next two internal rows, and escaping on the bottom layer, then you could completely route a 64-pin BGA on the top and bottom PCB layers. Typically, the power and ground pins in the BGA package are located in the center of the array. Where engineers use a multilayer PCB, they usually connect power and ground pins to the internal power and ground layers. A multilayer PCB dramatically increases the manageable number of pins.

When you select a routing scheme, you must consider the impact on board space. For example, if you are routing signal pins to the top layer that normally would have gone to an internal layer, then this increases your device's total board real estate. In addition, if you use through-hole vias integral to all the pads, you'll effectively prevent the mounting of components (discretes) on the bottom side of the board below the BGA. When you are using two array devices, you should carefully plan the device pinout and motherboard routing; this way you may be able to place all the devices on opposite sides of the board. This minimizes signal lengths and simplifies the routing complexity between them.

## Surface Mount Assembly

You can use existing surface mount equipment to place the BGA on circuit boards. The capital cost required to manage the BGA package is low when you compare it with other high pin-count packages. The BGA is compatible with existing handling media. Also, trays, tapes, reels, etc., are available for most BGA body sizes. You'll find that the package is far more robust than peripherally leaded devices (which dramatically reduces handling damage).

Before assembly, you must apply flux (paste or liquid) to the pads on the PCB in order to reduce the amount of oxide on both the pads and the solder balls. With BGAs, you need not add solder volume to the joint (e.g., by applying solder paste), because the solder ball is composed of near-eutectic solder and the entire volume of the ball is melted during the reflow process. A solder ball with 30 mil diameter provides enough solder volume to give an 18 to 22 mil standoff, depending upon the package size and solder ball array layout. This is enough standoff to ensure solder-joint integrity, even if the device or PCB warps at the elevated reflow temperature. If the standoff needs to be increased by one or two mils, you should apply an amount of solder paste equal to 14% of the ball volume. (For example, with an 8 mil stencil and 25 mil apertures, half the solder paste volume will be deposited.) If you apply more solder paste than recommended, you increase the possibility of joint voiding, especially if you are using fast oven ramp rates and volatile flux. Voids are formed when the hot flux volatilizes and is trapped within the joint. Voids form at the bottom of the ball (motherboard interface) but travel to the joint and package interface because of buoyancy effects.

Device placement with BGAs is relatively simple compared with fine pitch QFPs. Because of the tight ball pattern to device edge tolerance (better than  $\pm 3$  mils), you can place these devices with conventional accuracy and resolution using the body outline. Machines used for placing small components (chip shooters) use this placement technique. While BGA is compatible with standard placement equipment, equipment designed specifically for the placement of BGAs is now on the market. This equipment employs upward looking lasers which center to the ball array itself. Some equipment can also check for missing balls and measure coplanarity in real-time. The BGA is self-centering during the reflow process; thus, perfect alignment will occur even if the device balls are out of alignment with the pads by 50%. As the balls melt, the surface tension of the solder balls draws the device into alignment.

The reflow process may occur in IR, convection or mixed heating ovens. This process is very similar to the one you used with leaded surface mount devices. Since the solder balls start with a diameter of about 25 mils and collapse to between 3 to 7 mils during reflow, this process is sometimes referred to as the Controlled Chip Carrier Collapse Connection (C5). You must expose each solder joint to the 2%Ag eutectic liquidus temperature of 178°C immediately after the high temperature pre-soak period where the flux is mobile and active. The pre-soak temperature is defined by the type and brand of flux used. Keep these important constraints in mind:

- 210°C to 215°C is the optimal temperature range,
- keep the maximum temperature below 240°C, and
- use above liquidus temperature for 60 to 90 seconds.

When the BGA is reflowing, the joints are heated more from the board and the device than from direct air impingement or IR exposure to the leads. When you profile the furnace for the BGA, position the thermocouples beneath the package—preferably within an actual solder joint. You will need thermocouple wire that is thin enough to solder the device to the board normally without tilt or non-contacting balls. If one thermocouple wire is used, then you should place the wire bead under the package's center ball (minimum peak temperature). You should also position a thermocouple under an outermost ball (maximum peak temperature). Typically, you will then hand-place the package onto a board footprint, secured with KAPTON tape. Then you reflow the package. The thermocouple(s) should now be soldered within the desired solder joint(s). With the proper care, you may use this apparatus repeatedly to profile the reflow furnace.

Currently, the JEDEC standard for coplanarity of BGA devices is 6 mils, regardless of package size or pin-count. Coplanarity is defined as the maximum distance from the bottom of the highest ball to a seating plane formed by the three balls the package would rest on if it were placed on a perfectly flat surface. BGA coplanarity is primarily affected by:

- Warpage of the overmolded substrate, and
- Tip height of the solder balls

Warpage of the substrate is the dominant factor, since the solder ball heights are relatively uniform. The 225-pin PBGAs have a worst case coplanarity of 4 mils. Studies show that packages can be mounted successfully with even a 10 mil coplanarity.

## Solder Joint Inspection

Because joints are not easily visible in a BGA package, solder joint inspection is made more difficult than with some other surface mount packages. This drawback is offset by significantly higher assembly yields. Because of the atomic density of the lead in the solder joint, standard real-time X-ray systems may only be useful in detecting shorts and missing or double bonds. You will need higher resolution systems to observe more subtle anomalies such as voids, total wetting of the motherboard pad, and solder splattering or balling. You may also use low cost X-ray systems with the ability to detect voids and non-contact balls.

As we mentioned earlier, a simple modification to the shape of the solder pad allows inexpensive X-ray systems to detect opens and verify solder pad wetting. A tab (or X-ray bump) is placed on the side of the round solder mask defined pad. During reflow, if the joint is sound, then solder from the ball will fill the tab area.



*If you are using solder plated motherboards, then the tab would already be filled with solder which was stenciled on. In this case, the X-ray equipment must be of sufficient resolution to distinguish between stenciled-on solder and solder from the device.*

## F.3 Rework and Repair

Since BGA assembly yields are significantly higher than for high pin-count leaded devices, the frequency of rework is reduced. With this in mind, you can determine when it is more cost effective to rework a board rather than to scrap it.

The costs of rework equipment for BGAs vary greatly with features included with the equipment. Some useful features are:

- selectable top and/or bottom heating,
- selectable IR and/or forced air heating,
- nitrogen blanket,
- auto-profiling capabilities,
- Split-prism optics manual placement,
- automated vision placement, and
- device removal head, or vacuum tool.



BGA devices are removed by heating all the solder joints above the 178°C liquidus temperature. If the removed device is being scrapped, then it is possible to disregard ramp-up rates and time over the liquidus temperature. Applying liquid flux beneath the device before heating and removal facilitates uniform heating and reduces motherboard oxidation for subsequent soldering processes. If the board has been removed from drypack conditions for more than 24 hrs, delamination of the die attach (popcorning) may occur. To avoid popcorning, bake the board for 24 hrs at 125°C before removing the device.

If other surface mount devices that are close to the device being removed are also subject to popcorning, then you must be careful not to cause delamination damage to them. To minimize temperature spillover, maintain the nozzle on the rework station 100 mils above the surface of the BGA, and apply heat from the top. The size of the nozzle should be less than or equal to the plastic mold on the device, since reflow of the BGA depends upon heat conduction through the body of the device. You may apply heat to the bottom but (unless you are extremely careful) this might lead to delamination damage and partial reflowing of adjacent devices.

Board damage can also occur if the solder mask is subjected to excess heat or if the solder mask has poor initial adhesion to the pad. This can produce significant damage to a board which uses solder mask defined pads. In addition, if the pads contain integral offset vias, then the thin strand of solder mask between the pad and the via may be subject to lifting. Therefore, to prevent damage to the board or other devices, you must be extremely careful in controlling the quantity of heat that you apply, the direction of the heat, and the amount of heat "spillover".

After you remove the device, a large amount of solder will remain on the pads. You can remove this with a solder sucker, or solder wick. If you do not remove the excess solder, it will interfere with the self-alignment of the replacement BGA package. Apply new flux to the pads before replacing the device. Usually, you won't re-apply solder paste, because adjacent devices interfere with the stencil. If a placement machine with vision is not available, you may replace the device by hand. A silk screened outline of the package body on the PCB will help the manual alignment process. Split-prism optics allow you to simultaneously view the solder bumps and the pads. This can facilitate placement. The reflow profile used for rework should match the initial reflow profile as much as possible. During the replacement reflow, be careful not to damage the board and adjacent devices (see above).

If you carefully avoided delamination damage during the removal process, then you will usually be able to retest the BGA device. You might even be able to rework it. If the balls are approximately the same size (and are approximately round), you can test the device after cleaning the balls with a flux cleaner. Most BGA packages use a pogo-pin design which allows testing if there are small, or even missing, balls. You might have to use a flux/reflow/clean cycle if the balls becomes oblong.

Rebumping the array is NOT recommended unless the process is thoroughly qualified. To rebump the array, you first remove the solder from each pad. If robotic or manual bumping equipment is not available then individual 30 mil balls must be fluxed and placed on the pads using tweezers, then reflowed. This manual operation is very labor intensive and you will have to compare the cost of rebumping with the original cost of the device.

## **F.4 Moisture Effects and Handling**

The BGA is susceptible to the effects of moisture which may delaminate the die from the substrate. The die attach epoxy can absorb and trap moisture. When the device is subjected to the reflow temperatures, the trapped moisture vaporizes and creates large internal stresses in the attach epoxy. The water vapor forms a bubble in the substrate beneath the die; this leads to delamination. If there is a large amount of trapped moisture, delamination will be violent (popcorning) and the delamination will extend to the outer perimeter of the substrate.

The BGA is currently specified to meet Level 3 of the JEDEC classifications for moisture sensitivity as proposed in specification A112. Level 3C states that the device will not exhibit delamination after exposure to 30°C/60%Rh for 48 hrs. Tundra Semiconductor Corporation recommends that you do not leave the device out of drypack for more than 12 hrs before mounting; otherwise, you should bake the device at 125°C for 24 hrs before mounting. This bakeout should be performed in an inert atmosphere such as nitrogen.

## **F.5 Conclusion**

Tundra carefully considered whether to adopt BGA packaging. We quickly realized that using BGA would be beneficial for our customers and ourselves. Our Universe clients are reaping the space-saving benefits of BGA packaging of our VME to PCI bridge, the Universe™.

Appendix G

Operating and Storage Conditions

Table G.1 : Recommended Operating Conditions

DC Supply Voltage ( $V_{DD}$ )	5 V
Ambient Operating Temperature ( $T_A$ Commercial)	0°C to +70°C
Power Dissipation (32-bit PCI)	3.35W
Power Dissipation (64-bit PCI)	3.75W

Table G.2 : Absolute Maximum Ratings

DC Supply Voltage	-0.5 to 6.0 V
Input Voltage ( $V_{IN}$ )	-0.5 to $V_{DD}+0.5$ V
DC Current Drain per Pin, Any Single Input or Output	±50 mA
DC Current Drain per Pin, Any Paralleled Outputs	±100 mA
DC Current Drain $V_{DD}$ and $V_{SS}$ Pins	±75 mA
Storage Temperature, ( $T_{STG}$ )	-65°C to +150°C



*WARNING: Stresses beyond those listed above may cause permanent damage to the devices. These are stress ratings only, and functional operation of the devices at these or any other conditions beyond those indicated in the operational sections of this document is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.*



Appendix H

Mechanical and  
Ordering Information

H.1 Mechanical Information

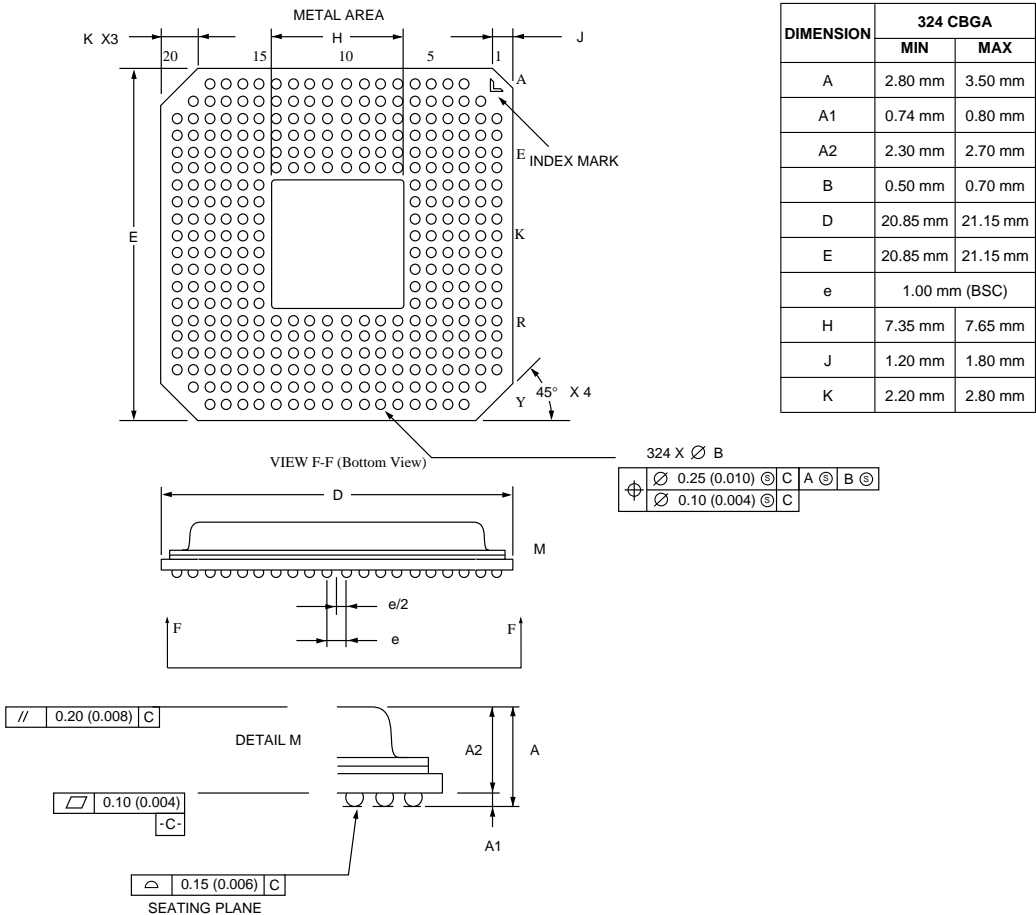
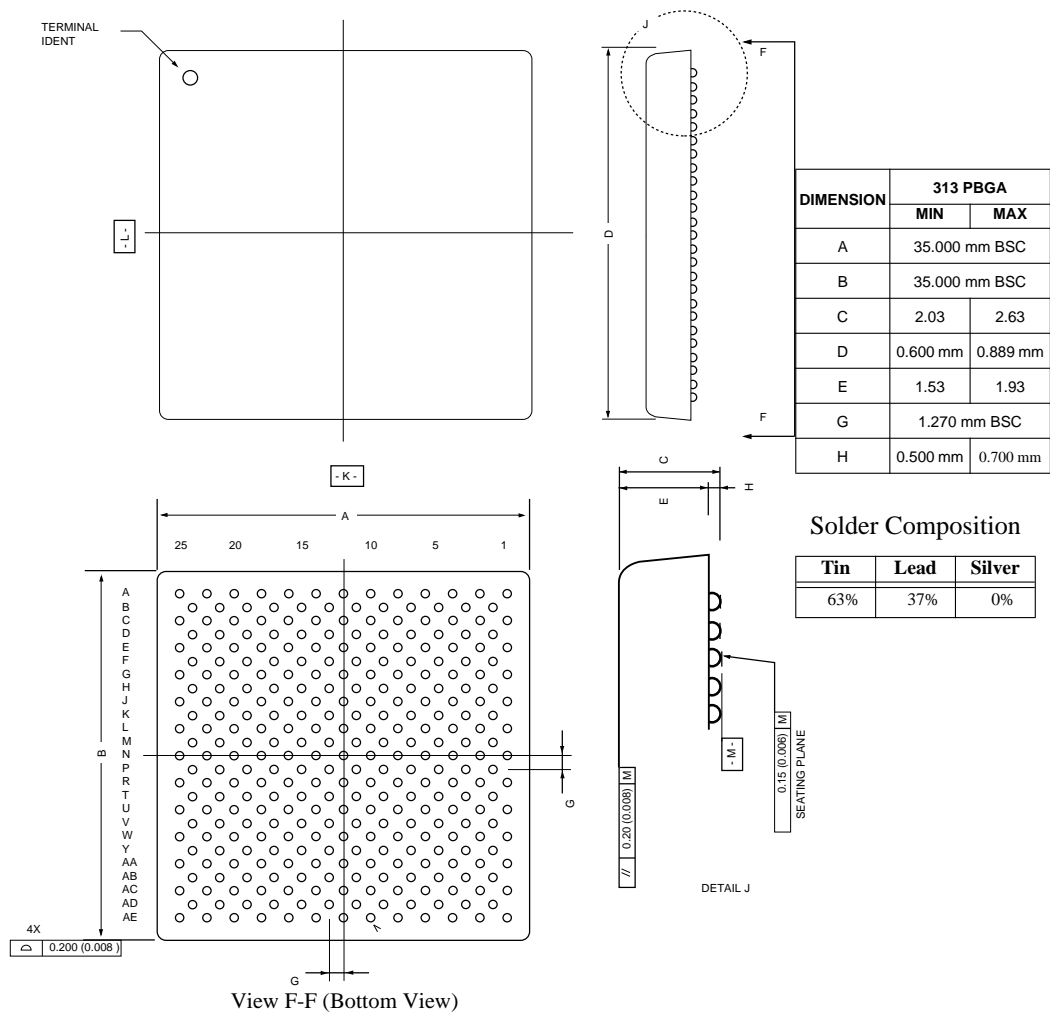


Figure H.1 : Mechanical Dimensions for the 324-Pin Ceramic BGA Package



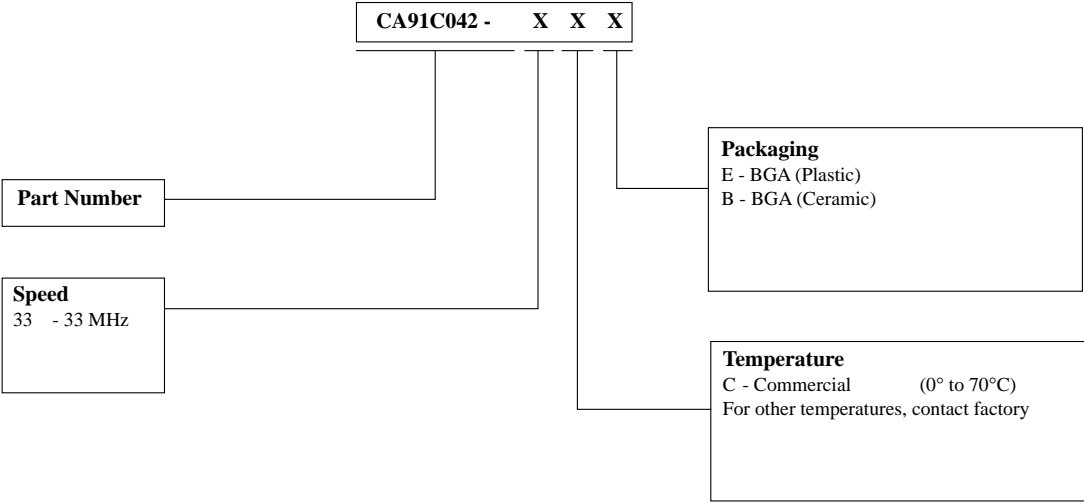
**Figure H.2 : Mechanical Dimensions for 313-Pin Plastic BGA Package**



*Note that dimensions and tolerancing for all mechanical drawings follow ANSI Y14.5M, 1982.*

## H.2    Ordering Information

Tundra Semiconductor Corporation products are designated by a Product Code. When ordering, refer to products by their full code. For detailed mechanical drawings or alternative packaging requirements, please contact our factory directly







# Index

## A

- Address Translation
  - PCI to VME 2-48
  - VME to PCI 2-46
- Addressing Capabilities
  - PCI master interface 2-31
  - VMEbus master interface 2-9
- ADOH Cycles
  - generating 2-41
- Architectural Diagram 2-2
- Auto Slot ID
  - proprietary method 2-21
  - VME64 specified 2-20

## B

- BCLR\* 2-8
- BI-Mode 2-24
- BSDL 2-105
- Bus Errors
  - DMA controller 2-72, 2-73, 2-86
  - IACK cycle 2-62
  - parity 2-29
  - prefetched reads 2-17
  - RXFIFO posted writes 2-16
  - TXFIFO posted writes 2-43

- Bus Ownership 2-82

## C

- CLK64 2-105
- Configuration Cycles 2-27, 2-31, 2-34, 2-44, 2-47, 2-49, 2-68, 2-87, 2-88
- Coupled Transactions
  - consecutive on PCI bus 2-37
  - error handling 2-52
  - PCI slave channel 2-37
  - retries on PCI bus 2-37
  - VMEbus slave channel 2-13
- Cycle Terminations
  - PCI master interface 2-32
  - PCI slave channel 2-43
  - VMEbus master interface 2-12
  - VMEbus slave channel 2-14, 2-16, 2-17, 2-19

## D

- D\_LLUE Register App A-46
  - UPDATE bit 2-80
- Data Packing/Unpacking
  - PCI slave channel 2-36
  - VMEbus master interface 2-11
- Data Transfer
  - PCI master interface 2-31
  - PCI slave channel 2-34
  - VMEbus master interface 2-10
  - VMEbus slave channel 2-13, 2-14, 2-16, 2-18

- Data Width 2-69
  - DCPP Register App A-43
    - DCPP field 2-70
  - DCTL Register
    - L2V bit 2-68
    - LD64EN bit 2-70
    - PGM field 2-68
    - SUPER field 2-68
    - VAS field 2-68
    - VCT bit 2-69, 2-70
    - VDW bit 2-70
  - DGCS Register App A-44
    - ACT bit 2-69, 2-71, 2-75, 2-79, 2-86
    - CHAIN bit 2-69
    - DONE bit 2-71, 2-72, 2-75, 2-76, 2-80, 2-81, 2-86
    - GO bit 2-70, 2-71, 2-72, 2-75, 2-79, 2-80, 2-81, 2-86
    - HALT bit 2-72, 2-79, 2-86
    - INT\_DONE bit 2-66
    - INT\_HALT bit 2-66
    - INT\_LERR bit 2-54, 2-55, 2-66, 2-73, 2-75, 2-78, 2-85, 2-86
    - INT\_M\_ERR bit 2-66, 2-85
    - INT\_STOP bit 2-66, 2-73, 2-75, 2-78, 2-85
    - INT\_VERR bit 2-54, 2-55, 2-66, 2-73, 2-75, 2-78, 2-85, 2-86
    - P\_ERR bit 2-71, 2-73, 2-75, 2-86
    - STOP bit 2-69, 2-71, 2-72, 2-75, 2-76, 2-80, 2-86
    - STOP\_REQ bit 2-72, 2-75, 2-76, 2-80
    - VERR bit 2-75, 2-86
    - VOFF field 2-71, 2-75, 2-78, 2-83, 2-84
    - VON field 2-71, 2-72, 2-75, 2-78, 2-82, 2-83, 2-84
  - DLA Register 2-68, App A-41
  - DMA Channel
    - PCI requests 2-30
    - PCI to VME transfers 2-82
    - VME tot PCI transfers 2-84
    - VMEbus release 2-8
    - VMEbus requests 2-7
  - DMA Completion 2-72
  - DMA Controller
    - defined 2-5
    - Direct mode operation 2-73
    - error handling 2-54, 2-86
    - FIFO Operation and bus ownership 2-82
    - interrupts 2-85
    - linked-list operation 2-76
  - DMA Interrupts 2-85
  - DTACK\* 4-5
  - DTACK\* Rescission 2-19
  - DTBC Register App A-40
    - DTBC field 2-68
  - DVA Register 2-68, App A-42
- ## E
- Error Handling
    - coupled transactions 2-52
    - DMA controller 2-54
    - parity 2-55
    - posted writes 2-52
    - prefetched reads 2-54
- ## F
- FIFOs
    - RDFIFO 2-16
    - RXFIFO 2-14

TXFIFO 2-38  
 First Slot Detector 2-19  
 FIT rate App D-1

## H

High Impedance Mode 2-104

## I

IACKIN\* 2-21  
 IACKOUT\* 2-21  
 Interrupt Acknowledge Cycles  
   auto-ID 2-21  
   bus errors 2-62  
   STATUS/ID 2-62  
 Interrupt Channel  
   interrupt sources 2-56  
   VMEbus release 2-8  
   VMEbus requests 2-6  
 Interrupt Generation  
   PCI bus 2-57  
   VMEbus 2-59  
 Interrupt Handling  
   internal sources 2-63  
   PCI bus 2-61  
   VMEbus 2-61  
 Interrupter  
   defined 2-4

IRQ\* 2-56

## J

JTAG 2-103, 2-105

## L

L\_CMDERR Register App A-37  
   CMDERR field 2-16, 2-32  
   L\_STAT bit 2-16, 2-32  
   M\_ERR bit 2-16, 2-32  
 LAERR Register App A-38  
   LAERR field 2-16  
 LERR 2-32, 2-53  
 Linked-List Operation of DMA 2-76  
 LINT\_EN Register App A-47  
   ACFAIL bit 2-59  
   DMA bit 2-54  
   interrupt sources 2-58  
   L\_DMA bit 2-54  
   LERR bit 2-32  
   SW\_IACK bit 2-64, 2-65  
   SW\_INT bit 2-58, 2-64  
   SYSFAIL bit 2-59  
   VERR bit 2-12, 2-43  
   VIR1x bits 2-58  
   VMEbus interrupt inputs 2-61  
 LINT\_MAP0 Register App A-51  
   interrupt sources 2-58  
   VERR field 2-58  
   VIRQ7-1 2-58  
   VMEbus interrupt handling 2-62  
   VMEbus interrupt inputs 2-58  
   VMEbus ownership bit 2-58  
   VOWN 2-58  
 LINT\_MAP1 Register App A-52  
   ACFAIL field 2-58  
   DMA field 2-58

- interrupt sources 2-58
  - LERR field 2-58
  - SW\_IACK 2-65
  - SW\_IACK field 2-58
  - SW\_INT field 2-58
  - SYSFAIL field 2-58
- LINT\_STAT Register 2-53, 2-59, App A-49
  - ACFAIL bit 2-58
  - DMA bit 2-58
  - interrupt sources 2-58
  - LERR bit 2-58
  - Software interrupts 2-65
  - SW\_INT bit 2-65
  - SYSFAIL bit 2-58
  - VERR bit 2-58
  - VMEbus interrupt handling 2-62
  - VMEbus interrupt inputs 2-58
  - VOWN bit 2-58, 2-65
- LMISC Register App A-34
  - CRT field 2-37
  - CWT field 2-37
- Locks
  - PCI slave channel 2-41
  - VMEbus slave channel 2-18
- LSI0\_BD Register App A-15
- LSI0\_BS Register App A-14
- LSI0\_CTL Register App A-13
- LSI0\_TO Register App A-16
- LSI1\_BD Register App A-19
- LSI1\_BS Register App A-18
- LSI1\_CTL Register App A-17
- LSI1\_TO Register App A-20
- LSI2\_BD Register App A-23
- LSI2\_BS Register App A-22
- LSI2\_CTL Register App A-21
- LSI2\_TO Register App A-24
- LSI3\_BD Register App A-27
- LSI3\_BS Register App A-26
- LSI3\_CTL Register App A-25
- LSI3\_TO Register App A-28
- LSIn\_BD Registers
  - BD field 2-47
  - Power-up options 2-102
- LSIn\_BS Registers
  - BS field 2-47
  - Power-up options 2-102
- LSIn\_CTL Registers
  - EN bit 2-47
  - LAS field 2-47
  - PGM field 2-47
  - Power-up options 2-102
  - PWEN bit 2-38, 2-47
  - SUPER field 2-47
  - VAS field 2-47
  - VCT field 2-34, 2-47
  - VDW field 2-47
- LSIn\_TO Registers
  - TO field 2-47

# M

- MAST\_CTL Register    App A-65
  - BUS\_NO field    2-31, 2-34
  - MAXRTRY field    2-32
  - PABS bit
    - And DMA FIFO    2-84
    - Bus error handling    2-53
  - PABS field    2-15, 2-16, 2-17, 2-30, 2-38, 2-39
  - PWON field    2-8, 2-10
  - VOWN bit    2-7, 2-42, 2-65
  - VOWN\_ACK bit    2-7, 2-42
  - VREL bit    2-8
  - VRL field    2-7
  - VRM field    2-7
- Master-Abort
  - defined    2-28
- MISC\_CTL Register    App A-67
  - BI bit    2-24
  - ENGBI bit    2-24
  - RESCIND bit    2-19
  - SYSCON bit    2-22
  - V64AUTO bit    2-102
  - VARB bit    2-23
  - VARBTO field    2-23
  - VBTO field    2-24
- MISC\_STAT Register    App A-69
  - DY4AUTO bit    2-102
  - DY4AUTOID field    2-22
  - DY4DONE bit    2-22
  - LCLSIZE bit    2-102
  - MYBBSY bit    2-8
  - PKGSIZE bit    2-102
  - RXFE bit    2-32
  - TXFE bit    2-8

MTBF    App D-1

# N

- NAND Tree Simulation    2-104
- Normal Mode    2-104

# P

- Parity
  - error handling    2-55
  - PCI master interface    2-33
- Parity Checking
  - Universe capability    2-29
- PBGA    App D-2
- PCI Aligned Burst Size (PABS)    2-15, 2-16, 2-38
- PCI Cycle Types
  - Universe capability    2-31
- PCI Interface
  - 32-bit versus 64-bit    2-25
  - cycle types    2-27
  - defined    2-4
  - Universe as master    2-30
  - Universe as slave    2-33
- PCI Master Interface
  - cycle terminations    2-32
  - data transfer    2-31
  - parity    2-33
- PCI Requests
  - DMA Channel    2-30
  - VME Slave Channel    2-30
- PCI Slave Channel
  - ADOH cycles    2-41
  - coupled transactions    2-37
  - cycle terminations    2-43
  - data packing/unpacking    2-36
  - data transfer    2-34
  - locks    2-41

- posted writes 2-38
- read-modify-writes 2-40
- TXFIFO 2-38
- VMEbus release 2-8
- VMEbus requests 2-6
- PCI Slave Images
  - defined 2-47
- PCI Terminations
  - defined 2-28
- PCI\_BS Register App A-11
  - BS field 2-88
- PCI\_CLASS Register App A-9
- PCI\_CSR Register App A-7
  - D\_PE bit 2-33, 2-55
  - DEVSEL field 2-34
  - DP\_D bit 2-33
  - PERESP bit 2-33, 2-55
  - R\_MA bit 2-14, 2-32
  - R\_TA bit 2-14, 2-32
  - S\_SERR bit 2-34
  - S\_TA bit 2-43
  - SERR\_EN bit 2-34, 2-55
- PCI\_ID Register App A-6
- PCI\_MISC0 Register App A-10
- PCI\_MISC1 Register App A-12
- PLL\_TESTSEL 2-104
- Posted Writes
  - error handling 2-52
  - errors 2-16
  - PCI slave channel 2-38
  - VMEbus slave channel 2-14
- power consumption App D-1
- Power-up
  - options 2-99
  - register access 2-19
- Power-up Options
  - auto-ID 2-101
  - BI-mode 2-101
  - PCI bus width 2-101
  - PCI CSR image space 2-100

- PCI slave image 2-100
- SYSFAIL assertion 2-101
- VME CR/CSR slave image 2-100
- VME register access slave image 2-99
- Prefetched Reads
  - bus errors 2-17
  - error handling 2-54
  - VMEbus slave channel 2-16
- PWON App B-5

## R

- RDFIFO 2-16
- Read-Modify-Writes
  - PCI slave channel 2-40
  - VMEbus slave channel 2-18
- Register App A-94
- Register Access
  - at power-up 2-19
  - configuration space 2-88
  - CR/CSR access 2-92
  - from VMEbus 2-89
  - I/O space 2-88
  - memory space 2-88
  - VMEbus register access image 2-90
- Register Map App A-2
- Registers
  - defined 2-87
- Reliability App D-1
- Request Modes 2-7
- Resets 2-94
- RXFIFO 2-14

## S

- SCYC\_ADDR Register App A-30
  - ADDR field 2-40

SCYC\_CMP Register    App A-32  
     CMP field    2-40

SCYC\_CTL Register    App A-29  
     SCYC field    2-40

SCYC\_EN Register    App A-31  
     EN field    2-40

SCYC\_SWP Register    App A-33  
     SWP field    2-40

SLSI Register    App A-35  
     BS field    2-49  
     EN bit    2-50  
     LAS field    2-49  
     PGM field    2-50  
     PWEN bit    2-50  
     SUPER field    2-50  
     VDW field    2-50

Special PCI Slave Image  
     defined    2-49  
     memory mapping    2-51

STATID Register    App A-57  
     STATID field    2-60

STATUS/ID  
     provided by Universe    2-60

SYSICON    2-105

## T

Target-Abort  
     defined    2-28

Target-Disconnect  
     defined    2-28

Target-Retry  
     defined    2-28

Terminology    1-4

Thermal vias    App D-2

Time-Outs  
     VMEbus    2-24  
     VMEbus arbiter    2-23

TXFIFO    2-38

## U

USER\_AM Register    App A-70  
     USER1AM, USER2AM fields    2-9

## V

V\_AMERR Register    App A-91  
     AMERR field    2-12, 2-43  
     IACK bit    2-12, 2-62  
     M\_ERR bit    2-12, 2-43  
     V\_STAT bit    2-12, 2-43

V1\_STATID Register    App A-58

V2\_STATID Register    App A-59

V3\_STATID Register    App A-60

V4\_STATID Register    App A-61

V5\_STATID Register    App A-62

V6\_STATID Register    App A-63

V7\_STATID Register    App A-64

VAERR Register    App A-92  
     VAERR field    2-12, 2-43

VCSR\_BS Register    App A-95  
     BS field    2-92

VCSR\_CLR Register    App A-93  
     FAIL bit    2-97  
     SYSFAIL bit    2-21, 2-97

VCSR\_CTL Register    App A-89  
     EN bit    2-92  
     LAS bit    2-92

VCSR\_SET Register    App A-94  
     FAIL bit    2-97  
     SYSFAIL bit    2-21, 2-97

VCSR\_TO Register    App A-90  
     TO field    2-92

- VERR 2-32, 2-53
- VINT\_EN Register App A-53
  - interrupt sources 2-59
  - PCI interrupt inputs 2-61
  - V\_DMA bit 2-54
  - V\_LERR bit 2-16, 2-32
  - V\_VERR bit 2-12, 2-43
  - VSW\_INT bit 2-64
- VINT\_MAP0 Register App A-55
  - PCI interrupt inputs 2-60
- VINT\_MAP1 Register App A-56
  - interrupt sources 2-59
- VINT\_STAT Register App A-54
  - IACK cycle error 2-63
  - interrupt sources 2-59
  - PCI interrupt inputs 2-61
  - VMEbus interrupt handling 2-62
  - VSW\_INT bit 2-65
- VME Slave Images
  - defined 2-44
- VMEbus Arbitration 2-23
  - arbiter time-out 2-23
  - priority mode 2-23
  - round robin 2-23
  - single level 2-23
- VMEbus Interface
  - BI-mode 2-24
  - configuration 2-19
  - CR/CSR access 2-92
  - defined 2-3
  - first slot detector 2-19
  - requester 2-6
  - system clock 2-22
  - system controller 2-22
  - Universe as master 2-9
  - Universe as slave 2-12
  - VMEbus release 2-8
  - VMEbus time-out 2-24
- VMEbus Master Interface
  - addressing capabilities 2-9
  - cycle terminations 2-12
  - data packing/unpacking 2-11
  - data transfer 2-10
- VMEbus Register Access Image 2-90
- VMEbus Requester
  - demand mode 2-7
  - DMA channel 2-7
  - fair mode 2-7
  - interrupt channel 2-6
  - PCI slave channel 2-6
  - request levels 2-7
- VMEbus Slave Channel
  - coupled transactions 2-13
  - DTACK\* rescission 2-19
  - errors 2-16, 2-17
  - locks 2-18
  - PCI requests 2-30
  - posted writes 2-14
  - prefetched reads 2-16
  - RDFIFO 2-16
  - read-modify-writes 2-18
  - RXFIFO 2-14
- VMEbus System Controller
  - IACK daisy chain 2-23
  - VMEbus arbitration 2-23
- Vn\_STATID Registers
  - ERR bit 2-63
  - STATID field 2-62
- VOFF timer 2-72, 2-83, 2-84
- VON 2-71
- VRAI\_BS Register App A-88
  - BS field 2-19, 2-90
- VRAI\_CTL Register App A-87
  - EN bit 2-90
  - PGM field 2-90
  - SUPER field 2-90
  - VAS field 2-90



VSI0_BD Register	App A-73
VSI0_BS Register	App A-72
VSI0_CTL Register	App A-71
VSI0_TO Register	App A-74
VSI1_BD Register	App A-77
VSI1_BS Register	App A-76
VSI1_CTL Register	App A-75
VSI1_TO Register	App A-78
VSI2_BD Register	App A-81
VSI2_BS Register	App A-80
VSI2_CTL Register	App A-79
VSI2_TO Register	App A-82
VSI3_BD Register	App A-85
VSI3_BS Register	App A-84
VSI3_CTL Register	App A-83
VSI3_TO Register	App A-86
VSI <sub>n</sub> _BD Registers	
BD field	2-44
VSI <sub>n</sub> _BS Registers	
BS field	2-44
VSI <sub>n</sub> _CTL Registers	
EN bit	2-44
LAS field	2-44
LD64EN bit	2-16, 2-44
LLRMW bit	2-18, 2-44
PGM field	2-44
PREN bit	2-16, 2-44
PWEN bit	2-14, 2-44
SUPER field	2-44
VAS field	2-44
VSI <sub>n</sub> _TO Registers	
TO field	2-44
VSYSCLK	2-105

